Décision dans l'incertain

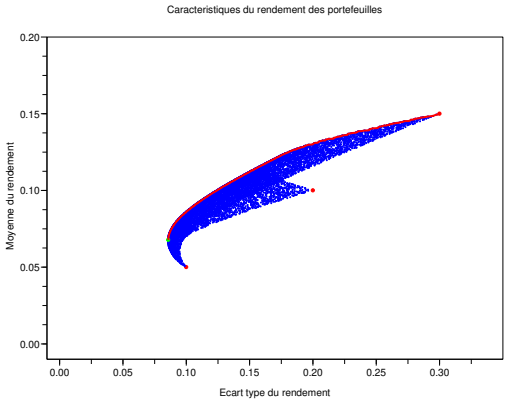
Ecole des Ponts, Avril/Mai 2020

JEAN-PHILIPPE CHANCELIER BERNARD LAPEYRE

Table des matières

Probabilité, Espérance, Variance

Choix optimal de portefeuille



Bernard Lapeyre

http://cermics.enpc.fr/~bl

Plan

FONCTIONNEMENT DU COURS

- Site du cours

  http://cermics.enpc.fr/~bl/decision-incertain
- 6 séances : 1h de cours, 1h30 de TP informatique
- une conférence qui remplace le dernier cours (le 29 mai)
- Enseignants (cours) :
  - Jean-Philippe Chancelier (jean-philippe.chancelier@enpc.fr)
  - Bernard Lapeyre (bernard.lapeyre@enpc.fr)
- Enseignants (TPs informatiques) :
  - Oumaïma Beincheikh (oumaima.bencheikh@enpc.fr)
  - Adel Cherchali (cherchaliadel@gmail.com)
  - Hachem Madmoun (hachem.madmoun@enpc.fr)
  - Sophian Mehalla (Sophian.Mehalla@milliman.com)
- Language de programmation: Python.
- [Scicoslab (Matlab clone) possible si souhaité.]

ORGANISATION ET RÈGLES DE VALIDATION

Organisation

- une semaine entre le cours et le TD correspondant ...
- ... pour: lire le cours, faire les exos, préparer le TP info
- une feuille d'exos par séance disponible sur le site

Évaluation

- présence vérifiée en cours et en TD
- pas de contrôle formel à l'issue du module
- ... **sauf** si plus d'une absence constatée (examen dans ce cas)
- rendu (individuel) de l'un des 5 TP informatiques rédigé en détails (LaTeX suggéré).

OBJET DU COURS

- Exposer des *situations* où la modélisation probabiliste est utile/indispensable pour prendre des décisions.
- Décrire un outil nouveau (les *chaînes de Markov*) et montrer comment cet outil peut être utilisé de façon effective.
- Implémenter les méthodes mathématiques proposées.

AUJOURD'HUI ...

- Des rappels de probabilité (espérance, variance, ...)
- Théorie du portefeuille de Markowitz.
  - Pourquoi des actifs (actions,...) ayant des rendements aux caractéristiques diverses peuvent ils coexister ?
  - Comment s'y prendre pour optimiser un portefeuille d'actif ?

LE RENDEMENT D'UN ACTIF

- Actif i (action, obligation, ...) de *rendement* R_i
- Rendement = sur une période de temps donnée T:

  1E à l'instant 0 va rapporter $(1+R_i)$E en T.
- Plusieurs actifs (ex CAC40 $i = 1 \dots 40$), vecteur de rendement $R = (R_1, \dots, R_n)$.
- Rendements *déterministes* et un *marché* où l'on peut acheter et vendre ces actifs,

  tous les R_i doivent être égaux.
- Mais c'est un fait d'expérience que les rendements des actifs n'ont pas des caractéristiques identiques: il est *nécessaire* de les supposer *aléatoires*: "Risk is not an add-on"[1]...
- ce qui nous amène à quelques rappels de probabilité ...

[1] Voir R.C.Merton interviewed in [Bernstein(2007)].

PROBABILITÉ ET ESPÉRANCE : RAPPELS

- Description d'un expérience aléatoire : une *probabilité*, sur $(\Omega, \mathcal{A})$, $\mathbb{P}$ qui opère sur des ensembles de $\mathcal{A}$.
- $A \in \mathcal{A}$, $0 \leq \mathbb{P}(A) \leq 1$ et $\mathbb{P}(\Omega) = 1$. $\sum_{i\geq 1} A_i$ signifie une réunion *dénombrable* et *disjointe*.

$$\mathbb{P}\left(\sum_{i\geq 1} A_i\right) = \sum_{i\geq 1} \mathbb{P}(A_i)$$

- À une probabilité $\mathbb{P}$ est associée une espérance $\mathbb{E}$ (qui opère sur des variables aléatoires à valeurs dans $\mathbb{R}$)
  - $\mathbb{E}(\mathbf{1}_A) = \mathbb{P}(A)$.
  - *linéarité* ($\mathbf{X}$ et $\mathbf{Y}$ positifs ou intégrables, $\mathbf{X}_i$ positifs).

$$\mathbb{E}(\lambda \mathbf{X} + \mu \mathbf{Y}) = \lambda \mathbb{E}(\mathbf{X}) + \mu \mathbb{E}(\mathbf{Y})$$

$$\mathbb{E}\left(\sum_{i\geq 1} \mathbf{X}_i\right) = \sum_{i\geq 1} \mathbb{E}(\mathbf{X}_i)$$

ESPÉRANCE ET VARIANCE

- L'espérance est *linéaire*.
- Variance : $\mathrm{Var}(\mathbf{X}) := \mathbb{E}\left\{(\mathbf{X}-\mathbb{E}(\mathbf{X}))^2\right\} = \mathbb{E}(\mathbf{X}^2) - \mathbb{E}(\mathbf{X})^2$.
- $\mathrm{Var}(\mathbf{X}) = 0$ implique $\mathbf{X} = Cte$ (p.s.).
- Covariance :
  $\mathrm{Cov}(\mathbf{X},\mathbf{Y}) = \mathbb{E}\left\{(\mathbf{X}-\mathbb{E}(\mathbf{X}))\,(\mathbf{Y}-\mathbb{E}(\mathbf{Y}))\right\} = \mathbb{E}(\mathbf{XY}) - \mathbb{E}(\mathbf{X})\mathbb{E}(\mathbf{Y})$.
- La linéarité de l'espérance permet de prouver

$$\mathrm{Var}\left(\sum_{i=1}^n \lambda_i \mathbf{X}_i\right) = \sum_{i=1}^n \sum_{j=1}^n \lambda_i\lambda_j \mathrm{Cov}(\mathbf{X}_i,\mathbf{X}_j)$$

- La variance fonctionne comme une forme *quadratique*, la covariance comme une *forme bilinéaire*.
- Cas particulier : si indépendance des $\mathbf{X}_i$, $\mathrm{Cov}(\mathbf{X}_i,\mathbf{X}_j) = 0$, $i \neq j$

$$\mathrm{Var}\left(\sum_{i=1}^n \mathbf{X}_i\right) = \sum_{i=1}^n \mathrm{Var}(\mathbf{X}_i).$$

VECTEUR ET MATRICE DE VARIANCE-COVARIANCE

- $\mathbf{X} = (\mathbf{X}_1,\ldots,\mathbf{X}_n)$ un vecteur de $\mathbb{R}^n$. Γ matrice de variance-covariance de $\mathbf{X}$

$$\Gamma = \left(\mathrm{Cov}(\mathbf{X}_i,\mathbf{X}_j)\right)_{1\leq i,j\leq n}$$

- Permet de calculer $\mathrm{Var}\left(\sum_{i=1}^n \lambda_i \mathbf{X}_i\right)$

$$\mathrm{Var}\left(\sum_{i=1}^n \lambda_i \mathbf{X}_i\right) = \lambda^T\Gamma\lambda = \sum_{i=1}^n\sum_{j=1}^n \lambda_i\lambda_j\Gamma_{ij}.$$

- Γ est une matrice symétrique, positive ($\lambda^T\Gamma\lambda \geq 0$, pour tout λ).

ESPÉRANCE ET VARIANCE COMME CRITÈRES DE DÉCISION

- Comment choisir entre 2 v.a. $\mathbf{X}$ et $\mathbf{Y}$ dont on ne connait que la loi ?
- On utilise (souvent) l'espérance et la variance comme critère de choix (plus l'espérance est grande ou plus la variance est petite, "meilleure" est la v.a.).
- C'est discutable [2], mais c'est l'approche la plus simple, bien justifiée dans le cas (restrictif) d'un vecteur gaussien.
- Des "mesures de risque" mieux fondées (Voir [Föllmer and Schied(2008)]) existent et sont aussi utilisées : "value at risk", "expected shortfall", ...
- ... au prix toutefois d'une complexité accrue de la modélisation et/ou des calculs.

[2] On peut avoir $\mathbb{P}(\mathbf{X} \leq \mathbf{Y}) = 1$, sans que $\mathbf{X}$ et $\mathbf{Y}$ soient comparables pour l'ordre partiel suggéré (voir la feuille d'exercice pour un exemple).

PORTEFEUILLE DE MARKOWITZ : LE MODÈLE

- d actifs (actions, obligations, ...). $\mathbf{R}_i$ le rendement, aléatoires, du $i$ème actif.

$$1\text{E en actif } i \text{ en } 0 \to (1 + \mathbf{R}_i)\text{E en } T.$$

- $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_d)$ le vecteur, aléatoire, des rendements
- $r = (\mathbb{E}(\mathbf{R}_1), \ldots, \mathbb{E}(\mathbf{R}_d))$, le vecteur des moyennes des rendements, par convention on suppose

$$r_1 \leq r_2 \leq \cdots \leq r_d.$$

- Γ la matrice de variance-covariance de $\mathbf{R}$.

$$\sigma_i^2 = \mathrm{Var}(\mathbf{X}_i) = \Gamma_{ii}.$$

PORTEFEUILLE DE MARKOWITZ : LES PORTEFEUILLES

- On va considérer des portefeuilles, composés de quantités d'actifs λ_i, de valeur initiale V_0 égale (par convention) à 1E

$$V_0 = \sum_{i=1}^{d} \lambda_i = 1, \quad V_T = \sum_{i=1}^{d} \lambda_i (1 + \mathbf{R}_i) = 1 + \sum_{i=1}^{d} \lambda_i \mathbf{R}_i.$$

- Souvent $0 \leq \lambda_i \leq 1$. Mais dans certains cas on acceptera des quantités négatives pour λ_i (emprunt d'actif).
- Le gain du portefeuille est égal à $\mathbf{G} = V_T - V_0$

$$\mathbf{G} = \sum_{i=1}^{d} \lambda_i \mathbf{R}_i$$

- La moyenne et la variance de $\mathbf{G}$ se calcule facilement

$$\mathbb{E}(\mathbf{G}) = \lambda.r \qquad \mathrm{Var}(\mathbf{G}) = \lambda^T \Gamma \lambda$$

en fonction de r et Γ. On va comparer les portefeuilles grâce à ces deux valeurs.

PORTEFEUILLE DE MARKOWITZ : LES 2 CRITÈRES

- L'hypothèse de base du modèle est que l'investisseur va classer les portefeuilles en utilisant la moyenne et l'écart-type (= la racine de la variance) du gain.
- Un portefeuille de moyenne inférieure et de variance supérieure ne sera jamais choisi par un investisseur rationnel (c'est réaliste, mais ça reste une hypothèse).
- On définit un ordre *partiel* sur les portefeuilles :

  $\mathbf{G}_1$ préférable à $\mathbf{G}_2$, ssi $\mathbb{E}(\mathbf{G}_1) \geq \mathbb{E}(\mathbf{G}_2)$ et $\mathrm{Var}(\mathbf{G}_1) \leq \mathrm{Var}(\mathbf{G}_2)$.
- On suppose que les actifs de base ne sont pas comparables pour cet ordre partiel, ce qui impose

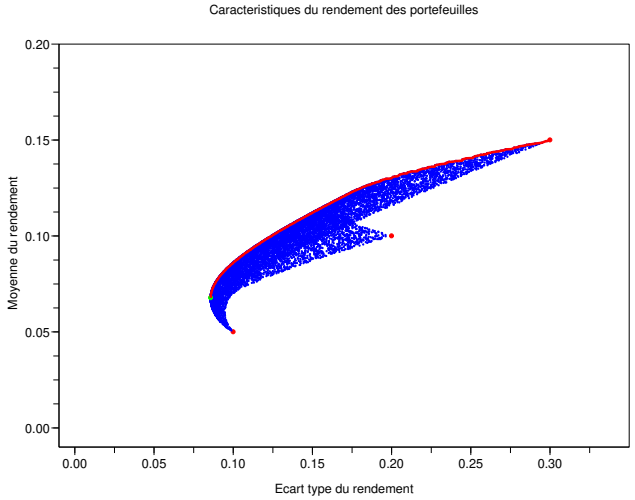
$$0 < \sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_d.$$

PORTEFEUILLE DE MARKOWITZ : FRONTIÈRE EFFICIENTE

- On s'intéresse aux portefeuilles "non dominés" (maximaux pour l'ordre partiel) : ceux sont les seuls susceptibles d'être utilisé par un intervenant rationnel.
- Ces portefeuilles "non dominés" forment une frontière de *Pareto* (ou d'indifférence).
- Markowitz l'appelle la *frontière efficiente*.
- On calculera cette frontière par exploration/simulation (en TD).
- Il existe d'autres méthodes plus efficaces (voir page 23) .

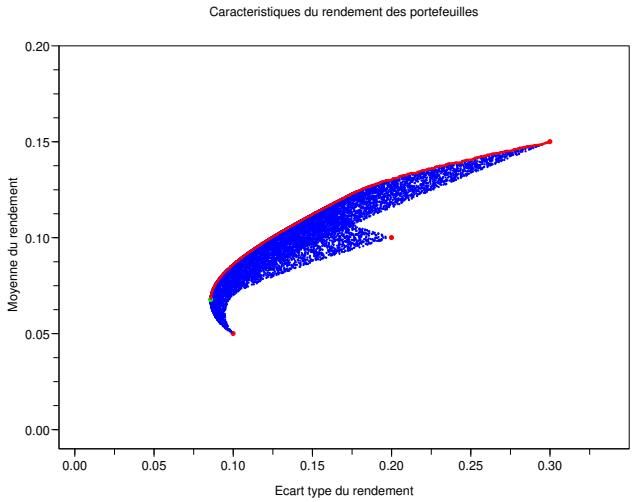
EXEMPLE DE FRONTIÈRE EFFICIENTE

- 3 actifs risqués, covariances nulles, frontière obtenue par simulation (cf TD), avec contraintes $0 \leq \lambda_i \leq 1$, $\lambda_1 + \lambda_2 + \lambda_3 = 1$.
- frontière efficiente : points non dominés.



FRONTIÈRE EFFICIENTE : REMARQUES

- Le point de variance minimale.
- Effet de diversification : portefeuille de variance inférieure à l'actif de variance minimale, rendement meilleur.
- Sur la frontière efficiente on choisit un compromis espérance-variance.

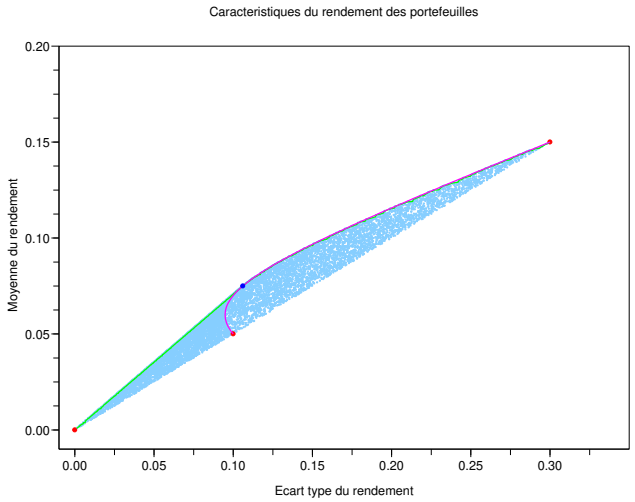


ACTIF SANS RISQUE ET PORTEFEUILLE DE MARCHÉ

- On suppose souvent l'existence d'un actif 0 sans risque (de variance nulle $\sigma_0 = 0$) et (donc) de rendement constant r_0.
- Cet actif correspond à un placement (quantité positive) ou un emprunt (quantité négative) dans une banque.
- Le rajout de cet actif modifie la forme de la frontière efficiente (cf TD) et fait apparaître un portefeuille remarquable: *le "portefeuille de marché"*.

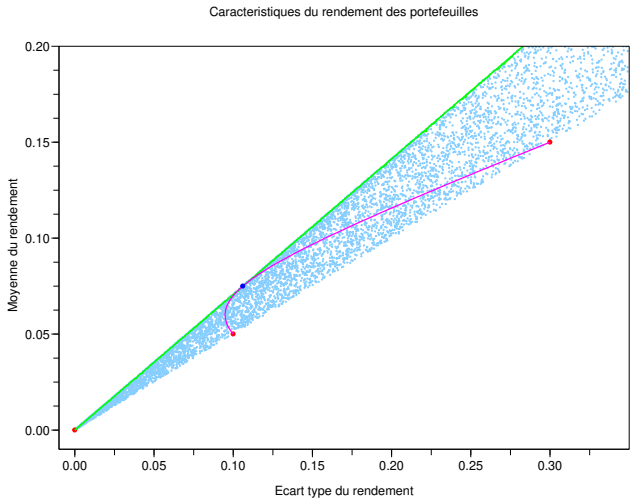
LE PORTEFEUILLE DE MARCHÉ

- Cas 2 actifs risqués et un actif sans risque $r_0 = 0$, frontière obtenue par simulation (cf TD), avec contrainte $\lambda_i \geq 0$.
- Frontière efficiente pour les deux actifs risqués, les trois actifs.
- Portefeuille de marché : le point où les deux frontières se séparent.



PORTEFEUILLE DE MARCHÉ

- Son intérêt apparaît lorsque l'on autorise l'*emprunt* de l'actif non risqué.
- La frontière efficiente est modifiée (cf TD). La droite se prolonge au delà de P.
- Cette droite porte le nom de "droite de marché".



Intérêt du portefeuille de marché

- La droite de marché domine strictement la frontière efficiente "sans emprunt" (sauf pour le portefeuille de marché où elle coïncide).
- Le seul point "rationnel" de la frontière efficiente, qui ne contient pas d'actif sans risque, est le portefeuille de marché.
- Si l'on peut emprunter, les portefeuilles "rationnels" sont obtenus par combinaison de l'actif sans risque et du portefeuille de marché.
- En empruntant, on peut obtenir un rendement de même moyenne mais de variance inférieure à celle du meilleur actif individuel.
- En empruntant, on peut obtenir un rendement de même variance mais de moyenne supérieure à celle du meilleur actif individuel.
- Si l'on peut emprunter, il faut le faire !
- Le portefeuille de marché fait intervenir l'ensemble des actifs risqués (tous les actifs peuvent coexister).
- Variance nulle : il faut tout investir dans l'actif sans risque.

Conclusions : finance et aléas

- Sans hasard, pas d'activité financière.
- ... ou dès qu'il y a du hasard, l'activité financière se développe.
- cf développement des produits dérivées suite à l'abandon en 1971 de la convertibilité du dollars en or (accord de Bretton-Wood).

Le portefeuille de marché comme solution d'un problème d'optimisation

- Pour calculer P il faut résoudre le problème d'optimisation suivant : max en λ (vérifiant les contraintes) de la pente du point $(\sqrt{\mathrm{Var}(\mathbf{G})}, \mathbb{E}(\mathbf{G}))$ ou encore

$$\max_{\lambda} \frac{\mathbb{E}(\mathbf{G})^2}{\mathrm{Var}(\mathbf{G})} = \max_{\lambda} \frac{(r^T\lambda)^2}{\lambda^T\Gamma\lambda}$$

- Le quotient $\frac{\mathbb{E}(\mathbf{G})}{\sqrt{\mathrm{Var}(\mathbf{G})}}$ s'appelle le *ratio de Sharpe* du portefeuille[3].

[3] William Sharpe a été lui aussi prix Nobel d'économie en 1990 avec Harry Markowitz et Merton Miller.

Calcul du portefeuille de marché

$$\max_{\lambda} \frac{\mathbb{E}(\mathbf{G})^2}{\mathrm{Var}(\mathbf{G})} = \max_{\lambda} \frac{(r^T\lambda)^2}{\lambda^T\Gamma\lambda}$$

- Voir cours optimisation de 1A.
- Lorsque l'on ne tient pas compte des contraintes de positivité sur λ, mais seulement de $\sum_{i=1}^n \lambda_i = \mathbf{1}^T\lambda = 1$, on peut résoudre ce problème.
- On peut utiliser un algorithme générique d'optimisation de type gradient. Ce type de problème est accessible dans `Scicoslab` ou `Python`, via une fonction d'optimisation. Voir le corrigé du TD.
- Le problème évoqué peut être traité explicitement, en utilisant l'inégalité de Cauchy-Schwartz.

CALCUL DU PORTEFEUILLE DE MARCHÉ

- En utilisant l'inégalité de Cauchy-Schwartz pour la norme $\sqrt{x^T\Gamma x}$, on obtient

$$r^T\lambda = (\Gamma^{-1}r)^T\Gamma\lambda \leq \sqrt{r^T\Gamma^{-1}r}\,\sqrt{\lambda^T\Gamma\lambda}.$$

On en déduit que

$$\sup_{\lambda,\sum_{i=1}^d \lambda_i=1} \frac{r^T\lambda}{\sqrt{\lambda^T\Gamma\lambda}} \leq \sqrt{r^T\Gamma^{-1}r}$$

et que l'égalité est atteinte pour $\lambda = \Gamma^{-1}r/(\mathbf{1}\,\Gamma^{-1}r)$ où $\mathbf{1} = (1,\ldots,1)$

PARAMÈTRISATION DE LA FRONTIÈRE EFFICIENTE PAR DES PROBLÈMES D'OPTIMISATION

- Une façon de calculer la frontière efficiente est de minimiser la variance à espérance égale à α, puis de faire varier α.

$$\min_{\lambda,\ \mathbf{1}^T\lambda=1,\ r^T\lambda=\alpha} \lambda^T\Gamma\lambda.$$

- Toujours en ignorant les contraintes de positivité sur λ, on peut encore résoudre ce problème
- Soit en utilisant une routine d'optimisation en éliminant les 2 contraintes (voir TD).
- Soit une utilisant deux multiplicateurs de Lagrange, le problème se résolvant alors de façon explicite en utilisant l'inverse de la matrice Γ (exercice).

CONCLUSIONS : DÉCIDER DANS L'INCERTAIN

- Décider en environnement incertain suppose d'*identifier* un modèle et de *choisir des critères* de décision (ce qui n'est pas toujours simple).
- Un fois ceci fait, on utilise des algorithmes pour résoudre les problèmes d'*optimisation* (ce qui n'est pas toujours simple).
- Une modélisation aléatoire simple peut éclairer la pratique et conduire à des règles opératoires.

HARRY MARKOWITZ

- Né en 1927, prix Nobel d'Economie 1990.
- Article fondateur, à 25 ans, en 1952 ("Portfolio Selection", The Journal of Finance).

- " ... when I defended my dissertation as a student in the Economics Department of the University of Chicago, Professor Milton Friedman argued that portfolio theory was not Economics, and that they could not award me a Ph.D. degree in Economics for a dissertation which was not in Economics. ... at the time I defended my dissertation, portfolio theory was not part of Economics. But now it is." Nobel Lecture, December 7, 1990, Harry M. Markowitz.

POUR ALLER PLUS LOIN : COURS DE L'ÉCOLE DES PONTS

- 1A : cours de "Probabilité", cours d'"Optimisation"
- 2A : cours "Processus Aléatoires", cours de "Recherche Opérationnelle", cours "Optimisation", cours "Méthodes Mathématiques pour la Finance", cours "Stratégie financière de l'entreprise"

BIBLIOGRAPHIE

P.L. Bernstein.
Capital Ideas Evolving.
Jhon Wiley and Son, 2007.

Hans Föllmer and Alexander Schied.
Convex and coherent risk measures.
http://www.alexschied.de/Encyclopedia6.pdf, 2008.

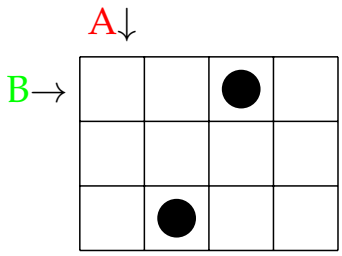
M. Haugh.
Asset allocation and risk management.
Available on ieor e4602 web page, Columbia University, 2009.

H.M. Markowitz.
Portfolio selection.
The Journal of Finance, 7(1):77–91, 1952.

H.M. Markowitz.
Portfolio Selection: Efficient Diversification of Investments.
John Wiley and Sons, 1959.

UN PUZZLE (DU SITE) DE LA NSA

- Deux œufs "au hasard" dans une matrice $p \times q$.
- Deux joueurs: le joueur A parcours la matrice en colonne, le joueur B en ligne.
- Le gagnant est celui qui atteint le premier l'un des deux œufs.



- Ce jeu est-il équitable ?
- La réponse dépend de p et q de façon bizarre:

$$\neq \text{ pour } (p=3,q=4),\ (p=4,q=4) \text{ et } (p=4,q=5)!$$

Cours de Décision dans l'incertain
Exercices : vendredi 3 avril 2020.

Exercice 1

1. Montrer que, si X est une variable réelle, $\mathbb{E}\left\{(X-\mathbb{E}(X))^2\right\} = \mathbb{E}(X^2) - \mathbb{E}(X)^2$.
2. Montrer que si $X = (X_1, \ldots, X_n)$ est un vecteur de $\mathbb{R}^n$ on a
$$\text{Var}\left(\sum_{i=1}^n \lambda_i X_i\right) = \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j \text{Cov}(X_i, X_j)$$
En déduire que la matrice symétrique $\Gamma = (\text{Cov}(X_i, X_j))_{1\leq i,j\leq n}$ est une matrice positive (i.e. $x.\Gamma x \geq 0$ pour tout $x \in \mathbb{R}^n$).
3. Montrer que la matrice de covariance est dégénérée (i.e. elle admet un noyau non réduit à $\{0\}$) si et seulement si le vecteur X prend ses valeurs dans un hyperplan affine strict de $\mathbb{R}^n$.
4. Soit ρ un nombre réel compris entre -1 et 1, à quelle condition sur ρ la matrice $n \times n$ suivante
$$\begin{pmatrix} 1 & \rho & \rho & \ldots & \rho \\ \rho & 1 & \rho & \ldots & \rho \\ \rho & \rho & 1 & \ldots & \rho \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \rho & \rho & \ldots & \rho & 1 \end{pmatrix}$$
peut elle être la matrice de covariance d'un vecteur aléatoire ?

Exercice 2 Soit un couple de variables aléatoires (X, Y) tel que $\mathbb{P}(X \leq Y) = 1$[1]. Si X et Y representent des rendements de portefeuille, laquelle de ces deux variables aléatoires vous paraît naturellement préférable ?

Donner un exemple de couple de variables aléatoires (X, Y) tel que $X \leq Y$ p.s., avec (bien sûr !) $\mathbb{E}(X) \leq \mathbb{E}(Y)$, mais telles que $\text{Var}(X) < \text{Var}(Y)$.

Ces deux variables aléatoires ne seront pas comparables pour l'ordre partiel sur les gains de portefeuille défini dans le cours[2].

En quoi cela questionne t'il le choix de l'ordre partiel introduit dans la modèlisation du portefeuille proposée par Markowitz ?

Exercice 3 Soit Γ une matrice symétrique définie positive. Montrer que $\phi(x, y) = x^T \Gamma y$ est un produit scalaire. On note $\|x\|_\Gamma = \sqrt{x^T \Gamma x}$ la norme associée.

1. Montrer que (Inégalité de Cauchy-Schwartz) :
$$|x^T \Gamma y| \leq \|x\|_\Gamma \|y\|_\Gamma$$
2. En déduire que :
$$r^T \lambda \leq \sqrt{r^T \Gamma^{-1} r} \times \sqrt{\lambda^T \Gamma \lambda}.$$
Puis que
$$\sup_{\lambda, \sum_{i=1}^d \lambda_i = 1} \frac{r^T \lambda}{\sqrt{\lambda^T \Gamma \lambda}} \leq \sqrt{r^T \Gamma^{-1} r}$$
et que l'égalité est atteinte pour $\lambda = \Gamma^{-1} r / (\mathbf{1}\, \Gamma^{-1} r)$ où $\mathbf{1} = (1, \ldots, 1)$

1. Remarquez que pour savoir si $\mathbb{P}(X \leq Y) = 1$, il faut connaître la loi du couple (X, Y).
2. Notez que cet ordre partiel ne suppose la connaissance que de la loi de X et de celle de Y (et non celle du couple (X, Y) comme précédemment).

Introduction à la théorie du portefeuille de Markowitz

Bernard Lapeyre
CERMICS, École des Ponts ParisTech

3 mars 2020

Table des matières

Le fichier source en `Scilab` correspondant à ce document est accessible. Il est partiellement mais pas totalement corrigé. Les partie à compléter sont signalées par le préfixe

`"QUESTION:"`.

Il vous revient de changer ces lignes. Le correction complète sera accessible à la fin du TD.

Le fichier `utils.sci` définit certaines primitives (graphiques pour l'essentiel) utilisées par la suite. Le télècharger et le sauvegarder sous le nom "utils.sci". Il devra être accessible par la suite dans votre directory de travail (qui s'obtient par "pwd" dans le shell de `Scicoslab`).

Introduction

On considère d actifs dont les rendements sont donnés pas $(R_1, \dots, R_d)$. L'hypothèse de rendement signifie que si on détient à l'instant 0 une quantité d'actif i de valeur V, la valeur de cette même quantité d'actif à l'instant T (égal à $T = 1$ an par exemple) sera donnée par $V(1 + R_i)$.

On suppose de plus que ces rendements ont des caractéristiques de moyenne et de variance connue. On note μ le vecteur des espérances $\mu_i = \mathbf{E}(R_i)$ et Γ la matrice de variance covariance, où $\Gamma_{ij} = Cov(R_i, R_j)$. On note $\sigma_i^2 = \mathrm{V}ar(R_i) = \Gamma_{ii}$.

1 Le cas à deux actifs risqués

On suppose que $d = 2$, que $\mu_1 = 5\%$ et $\mu_2 = 15\%$, que $\sigma_1 = 10\%$ et $\sigma_2 = 30\%$ et ρ étant un paramètre réel, Γ est donnée par

$$\Gamma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

1. Que représente ρ ? A quelle condition sur ρ la matrice Γ est la matrice de covariance d'un vecteur aléatoire ? Dans la suite, on prendra $\rho = 0$.

   On constitue un portefeuille de valeur initiale $X_0 = 1$ constitué d'une quantité x_1 d'actif 1 et x_2 d'actif 2 avec $x_1 \geq 0$, $x_2 \geq 0$ et $X_0 = x_1 + x_2 = 1$ (i.e. on répartit 1E entre le deux actifs risqués). On note X_T la valeur de ce portefeuille en T

   Vérifier que si le gain G_T est défini par $G_T = X_T - X_0$, $\mathbf{E}(G_T) = \mu_1 x_1 + \mu_2 x_2 = \mu_2 + x_1(\mu_1 - \mu_2)$ et $\mathrm{V}ar(G_T) = x.\Gamma x = \sigma_1^2 x_1^2 + \sigma_2^2(1-x_1)^2 + 2\rho\sigma_1\sigma_2 x_1(1-x_1)$.
2. Tracer les caractéristiques des actifs de base dans le plan moyenne,variance.

```
exec("utils.sci");// charge les primitives graphiques utiles

// On définit les caracteristiques des actifs
d=2;
mu=[0.05,0.15];
// Matrice de covariance: des 1 sur la diagonale, des rho ailleurs
rho=0.0;
covariance=rho*ones(d,d)+(1-rho)*eye(d,d);
sigma=[0.10,0.30];
Gamma = diag(sigma) * covariance * diag(sigma);

// Les caractéristiques des actifs de base
moyenne_actif=mu;
std_actif=sigma;

// Tracé des points représentant les actifs dans le plan
//                  (ecart-type,moyenne)
rectangle=plot_actifs(moyenne_actif,std_actif);
// On recupére le "rectangle" dans lequel on va dessiner par la suite.
```

3. Tracer la courbe $x_1 \in [0,1] \to (\mathbf{E}(G_T), \sqrt{\mathrm{V}ar(G_T)})$.

   Vérifier que l'on peut construire un portefeuille de même variance que l'actif 1 mais dont l'espérance du rendement est supérieure à celle de cet actif. Est il rationnel d'investir dans l'actif 1, si l'on cherche à minimiser son risque ?

   Quel sont les portefeuilles dans lesquels il paraît rationnel d'investir ?

```
// On parcours toutes les valeurs possibles de x_1 et x_2
// C'est facile lorsque d=2 ...
N=100;
x_1=[0:1/N:1];
x=[x_1;1-x_1];

moyenne_x=0;std_x=0;
for i=[1:N+1] do
  current_x = x(:,i);
  QUESTION: moyenne_x(i) = QUE VAUT LA MOYENNE DE CE PORTEFEUILLE
  QUESTION: std_x(i)= QUE VAUT L'ECART-TYPE DE CE PORTEFEUILLE
end;

plot2d(std_x, moyenne_x, style=1,rect=rectangle);
set_line(bleu,1);// on trace la courbe en bleu, épaisseur=1
```

4. Vérifier que l'on peut construire un portefeuille de variance minimum (et inférieure à celle de l'actif de variance minimum). C'est un exemple de l'*effet de diversification* dans la théorie des portefeuille.

```
// Calcul du point de variance minimum
[m,imin]=min(std_x);

plot2d(std_x(imin), moyenne_x(imin), rect=rectangle);// trace du point
set_dot(vert,4);// en vert, taille=4
```

5. Nous relaxons la condition $x_1 \geq 0$, $x_2 \geq 0$ tout en continuant à imposer $x_1 + x_2 = 1$ (la valeur totale de notre investissement initial reste égale à 1). Nous allons faire varier x_1 entre -10 et 0 (lorsque x_1 est négatif, on emprunte une quantité $|x_1|$ d'actif 1, mais la valeur totale du portefeuille doit toujours rester égale à 1).
   Tracer la courbe $x_1 \in [-5, 0] \rightarrow (\mathbf{E}(G_T), \sqrt{\mathrm{V}ar(G_T)})$. Vérifier que, si l'on accepte une variance grande, on peut constituer des portefeuilles d'espérance aussi grande que souhaitée (cet effet porte le nom d'*effet de levier* ou leverage effect). On comprend qu'il ne faille pas en abuser !

```
// On autorise l'emprunt de l'actif 1
N=1000;
moyenne_x=0;std_x=0;
for i=[0:N] do
  // quantité négative = emprunt de l'actif 1
  QUESTION: x_1= QUE VAUT x_1;
  x_2=1-x_1;
  x=[x_1;x_2];
  moyenne_x(i+1)=mu * x;
  std_x(i+1)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x,style=point, rect=rectangle);
set_line(vert,1);// trace les lignes en vert, taille=1
```

6. Tracer la courbe $x_2 \in [-5, 0] \rightarrow (\mathbf{E}(G_T), \sqrt{\mathrm{V}ar(G_T)})$. Vérifier que lorsque l'on emprunte l'actif 2 (x_2 négatif), l'on fait décroître l'espérance en augmentant la variance (ce qui est loin d'être optimal !).

```
// Que se passe t'il lorsque l'on emprunte de l'actif 2 ?
N=1000;
moyenne_x=0;std_x=0;
for i=[0:N]
  // quantité négative = emprunt de l'actif 2
  QUESTION: x_2= QUE VAUT x_2;
  x_1=1-x_2;
```

```
  x=[x_1;x_2];
  moyenne_x(i+1)=mu * x;
  std_x(i+1)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x,style=point, rect=rectangle);
set_line(rouge,1);// trace les lignes en rouge, taille=1
```

7. Recommencer l'expérience précédente avec des valeurs de ρ non nulle. Prendre par exemple

$$\rho = -0.5 \quad \text{et} \quad \rho = 0.5$$

Que peut représenter ces valeurs de ρ ?

```
// Matrice de covariance: des 1 sur la diagonale, des rho ailleurs
rho=0.5;// -0.5
covariance=rho*ones(d,d)+(1-rho)*eye(d,d);
Gamma = diag(sigma) * covariance * diag(sigma);
// etc ...
```

8. Nous allons introduire un nouvel actif, l'actif sans risque, qui comme son nom le suggère aura un rendement de variance nulle (ce qui implique que ce rendement n'est pas aléatoire). On supposera que ce rendement déterministe est inférieur à tous les rendements moyens des actifs risqués (pourquoi est-ce une hypothèse raisonnable ?). On prendra, ici, ce rendement égal à 0.

   On constitue des portefeuilles avec les 3 actifs (1 non risqué, 2 risqués) en tirant au hasard des coefficients (x_1, x_2, x_3) dans le simplexe $\{0 \leq x_i \leq 1, i = 1, 2, 3 \text{ et } x_1 + x_2 + x_3 = 1\}$. La fonction $simplexe(d)$ figurant dans `utils.sci` pour $d = 3$ fait ce travail.

   Matérialiser, en tirant un grand nombre points au hasard, la nouvelle frontière efficiente. Vérifier que :
   — la nouvelle frontière efficiente étend l'ancienne par de nouveaux points "non dominés" entre l'actif sans risque et un portefeuille tangent à l'ancienne frontière P.
   — que la variance reste bornée par la variance la plus grande (tant que l'on ne fait pas d'emprunt).

```
// On rajoute un actif sans risque de moyenne nulle
r0=0;
mu=[r0,mu];
// comme ce rendement est suppose deterministe, la matrice de
// variance covariance se complete par une ligne et une colonne de 0
QUESTION: Gamma= QUE VAUT GAMMA DANS CE CAS

moyenne_actif=mu;
std_actif=sqrt(diag(Gamma));

// On materialise les 3 actifs de base
plot2d(std_actif, moyenne_actif, style=1, rect=rectangle);
```

```
set_dot(noir,4);

// On considère des portefeuilles *avec l'actif sans risque*
// mais *sans emprunt*. On les tire au hasard dans le simplexe
// de dimension 3
N=1000;
moyenne_x=0;std_x=0;
for i=[1:N]
  x=simplexe(d+1);// tirage au hasard dans le simplexe
  moyenne_x(i)=mu * x;
  std_x(i)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x,style=-1, rect=rectangle);
set_dot(bleuclair,2);
```

Commentaire : On obtient de nouveaux points "non dominés" entre l'actif sans risque et un portefeuille tangent. La variance reste bornée par la variance de l'actifs de plus grande variance tant que l'on n'emprunte pas.

9. On va identifier un portefeuille particulier P, le "portefeuille de marché". P est le portefeuille correspondant au point de tangence de la droite passant par l'actif sans risque et de l'ensemble de tous les portefeuilles a coefficients positifs de la question précédente.

   Le point P est caractérisé par le fait qu'il maximise la pente des droites reliant le point $(\sigma_0 = 0, r_0 = 0)$ et les points correspondants à des portefeuilles y ne faisant pas intervenir d'actif sans risque.

   Toujours en procédant par simulation dans le simplexe, calculer P (en fait une approximation de P).

   Vérifier que le portefeuille P fait intervenir les 2 actifs risqués.

```
// On genere des portefeuilles sans actifs sans risque
N=1000;
moyenne_y=0;std_y=0;
for i=[1:N]
  QUESTION: y = LES PORTEFEUILLES SANS ACTIF SANS RISQUE

  // on calcule les moyennes et variances des portefeuilles y
  moyenne_y(i)=mu * y;
  std_y(i)=sqrt(y'*Gamma*y);
end;

// Le point P maximise la pente de la droite entre (sigma0=0, x_0=r0)
// et les portefeuilles y (sans actif sans risque)
r0=mu(1);
sigma0=0;// sigma0 = Gamma(1,1) = 0
pente=(moyenne_y - r0) ./ (std_y - sigma0);// calcul des pentes
[lambda,imax]=max(pente);

// Tracé du point P
x_P=moyenne_y(imax);
sigma_P=std_y(imax);
plot2d(sigma_P,x_P, style=1, rect=rectangle);
set_dot(vert,4);
```

```
// Tracé du segment "Actif sans risque -> P"
plot2d([sigma0,sigma_P],[r0,x_P], style=1, rect=rectangle);
set_line(vert,2);

// Tracé de la droite "actif sans risque -> P" au dela de P
lambda=(x_P-r0)/(sigma_P-sigma0);//pente de la droite
sigma=2.0;// arbitraire mais "grand"
x=r0+lambda*(sigma-sigma0);
plot2d([sigma0,sigma],[r0,x], rect=rectangle);
set_line(vert,2);
```

10. On autorise la détention d'une quantité de signe arbitraire d'actif sans risque (cela correspond soit à un emprunt, soit à un placement). Pour cela on vous suggère de tirer la quantité d'actif sans risque x_0 entre $[-4, 1]$ (on peut emprunter jusqu'à 4 fois ce que l'on possède). Puis on tire, les quantités d'actifs risqués uniformément sur le simplexe $\{x_1 + x_2 = 1 - x_0\}$.
    Tirer un grand nombre de portefeuille, calculer leurs moyennes et écarts-type, les tracer sur la figure.

```
// L'emprunt en actif sans risque est autorisé
N=1000;
moyenne_x=0;variance_x=0;
for i=[1:N]
  // On génère des portefeuilles dont la quantité
  // d'actif sans risque est uniforme sur [-4,1]
  x_0=grand(1,1,'unf',-4,1);
  s=simplexe(d);// tirage uniforme dans le simplexe de dim d.
  // On veut génèrer un portefeuille avec x_0 actifs sans risque
  // et de valeur totale x_0 + sum_{i=1,...,d} x_i = 1.
  QUESTION: x = COMMENT DEFINIR CE PORTEFEUILLE EN UTILISANT s

  moyenne_x(i)=mu * x;
  variance_x(i)=sqrt(x'*Gamma*x);
end;

// Tracé des points tirés au hazard
plot2d(variance_x, moyenne_x,style=-2,rect=rectangle);
set_dot(mauve,2);
```

    Vérifier que :
    — l'on obtient de nouveaux points "non dominés" au delà du portefeuille tangent.
    — le rendement (mais aussi la variance) peut devenir aussi grand que souhaité : *effet de levier*.
    — un emprunt permet de construire des portefeuilles dont la moyenne des rendements est plus élevée à variance égale : *emprunter permet d'augmenter le rendement*.
    — l'emprunt permet de construire des portefeuilles de même moyenne mais de variance inférieure : *emprunter permet de réduire le risque*. Il existe en particulier un portefeuille dont la variance est égale à celle de l'actif 2 (l'actif de rendement maximum) mais de rendement supérieur.
    — le seul point de la "frontière sans emprunt" qui n'est pas dominé par un point de la "frontière avec emprunt" est le point P : *si l'on ne souhaite pas emprunter, le seul*

point rationnel est P.

— le portefeuille P fait intervenir l'ensemble des actifs de base risqués (en dehors des actifs de base dominés par d'autres actifs de base).

11. On peut aussi autoriser la détention d'une quantité de signe arbitraire de l'un quelconque des actifs. Pour cela on tire "au hasard" les coefficients (x_1, x_2, x_3) du portefeuille sans imposer de condition de positivité. C'est un peut plus délicat puisque le support des tirages n'est plus compact et la loi uniforme n'a pas de sens. La fonction `arbitraire(d,alpha)` figurant dans `utils.sci` pour $d = 3$ fait ce travail. `alpha` est un paramètre réel à régler (`alpha` égal à 1 convient dans ce cas).

    Tirer un grand nombre de portefeuille, calculer leurs moyennes et écarts-type, les tracer. On constate que tous ces nouveaux portefeuilles restent en dessous de la droite de marché.

```
d=3;
N=1000;
moyenne_x=0;variance_x=0;
for i=[1:N]
  x=arbitraire(d);
  moyenne_x(i)=mu * x;
  variance_x(i)=sqrt(x'*Gamma*x);
end;

// Tracé des points tirés au hazard
plot2d(variance_x, moyenne_x,style=-2,rect=rectangle);
set_dot(mauve,2);
```

2 Le cas d'un nombre d'actifs risqués arbitraire

Lorsque $d > 2$ les phénomènes sont identiques mais moins explicites. On peut recommencer ce qui précède mais il faudra généraliser le choix de la matrice de variance covariance et procéder par simulation dans tous les cas. Notez que les programmes fournis en correction fonctionnent en dimension arbitraire (sauf aux questions 1 et 2).

A titre indicatif voici un exemple du cas $d = 3$.

1. Choix des actifs de base.

```
// Caracteristiques des actifs sans risque

d=3;rho=0.0;

min_esp=0.05;max_esp=0.15;
mu=[min_esp:(max_esp-min_esp)/(d-1):max_esp];

// On suppose que tous les actifs risqués ont une
// corrélation constante égale à ρ.
// On doit forcement avoir ρ >= -(1/(d-1)),
// sinon la matrice n'est pas une matrice de covariance (exercice!).
covariance= rho*ones(d,d)+(1-rho)*eye(d,d);

// On choisit un ecart type croissant en fonction de l'actif
min_sigma=0.1;max_sigma=0.3;
sigma=[min_sigma:(max_sigma-min_sigma)/(d-1):max_sigma];

// La matrice de variance covariance se calcule par :
Gamma = diag(sigma) * covariance * diag(sigma);

// Les caractéristiques des actifs de base
moyenne_actif=mu;
std_actif=sqrt(diag(Gamma));

// Tracé des actifs dans le plan (ecart-type,moyenne)
rectangle=plot_actifs(moyenne_actif,std_actif);
```

2. Tirages des portefeuilles à coefficients positifs.

```
// On simule des valeurs possibles
N=1000;
moyenne_x=0;std_x=0;
for i=[1:N]
  x=simplexe(d);
  moyenne_x(i)=mu * x;
  std_x(i)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x, style=1,rect=rectangle);
set_dot(bleu,2);

// Le point de variance minimum
[m,imin]=min(std_x);
plot2d(std_x(imin), moyenne_x(imin), rect=rectangle);
set_dot(vert,4);
```

3. On autorise l'emprunt de l'actif 1.

```
N=1000;
moyenne_x=0;std_x=0;
sigma_e=2;
for i=[1:N] do
  x=arbitraire(d,sigma_e);
  moyenne_x(i)=mu * x;
  std_x(i)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x,style=point, rect=rectangle);
f=gcf();Dessin=f.children(1).children(1).children(1);
set_dot(rouge,2);
```

4. On rajoute un actif sans risque de moyenne nulle.

```
r0=0;
mu=[r0,mu];
// comme le rendement est deterministe, la matrice de
// variance covariance se complete par
Gamma=[zeros(1,d+1);zeros(1,d)',Gamma];

// On materialise les 4 actifs de base
moyenne_actif=mu;
std_actif=sqrt(diag(Gamma));
plot2d(std_actif, moyenne_actif, style=1, rect=rectangle);
set_dot(noir,4);
```

5. On considère des portefeuilles *avec l'actif sans risque* mais sans emprunt. On les tire au hasard dans le simplexe.

```
N=1000;
std_x=0;moyenne_x=0;
for i=[1:N]
  x=simplexe(d+1);
  moyenne_x(i)=mu * x;
  std_x(i)=sqrt(x'*Gamma*x);
end;

plot2d(std_x, moyenne_x,style=-1, rect=rectangle);
set_dot(bleuclair,2);
```

6. Identification du portefeuille de marché P.

```
N=1000;
moyenne_y=0;std_y=0;
for i=[1:N]
  y = [0;simplexe(d)];    // un portefeuille sans actif sans risque
  moyenne_y(i) = mu * y;  // on calcule ses caractéristiques
  std_y(i)=sqrt(y'*Gamma*y);
end;

r0=mu(1);
sigma0=sqrt(Gamma(1,1));// sigma0 = Gamma(1,1) = 0
pente=(moyenne_y - r0) ./ (std_y - sigma0);// calcul des pentes
[lambda,imax]=max(pente);
x_P=moyenne_y(imax);
sigma_P=std_y(imax);
```

7. Tracé du portefeuille de marché et de la droite de marché associé.

```
// Tracé du point P
plot2d(sigma_P,x_P, style=1, rect=rectangle);
set_dot(red,2);

// Tracé du segment "Actif sans risque -> P"
plot2d([sigma0,sigma_P],[r0,x_P], style=1, rect=rectangle);
set_line(vert,2);

// Tracé de la droite "actif sans risque -> P" au dela de P
sigma=2.0;// arbitraire mais "grand"
lambda=(x_P-r0)/(sigma_P-sigma0);//pente de la droite
x=r0+lambda*(sigma-sigma0);
plot2d([sigma0,sigma],[r0,x],style=-1, rect=rectangle);
set_line(vert,2);
```

8. Avec actif sans risque et emprunt autorisé. On tire au hasard sans imposer le signe de l'actif sans risque. C'est fait par la primitive `arbitraire(d+1,sigma_e)` qui faitun choix (arbitraire) pour la loi de la quantité d'actif sans risque.

```
N=5000;
moyenne_x=0;variance_x=0;
for i=[1:N]
  x=arbitraire(d+1,sigma_e);
  moyenne_x(i)=mu * x;
  variance_x(i)=sqrt(x'*Gamma*x);
end;

// Tracé des points tirés au hazard
plot2d(variance_x, moyenne_x,style=-2,rect=rectangle);
set_dot(mauve,2);
```

Ce programme est paramétrable pour des valeurs de d et ρ arbitraires, si vous souhaitez expérimenter par vous même. Toutefois des problèmes d'échantillonage se pose lorsque d devient grand (au delà de 5, la loi uniforme sur le simplexe "a du mal à visiter les coins").

3 Solution des problèmes d'optimisation

1. Le calcul du portefeuille de marché P, s'exprime sous la forme d'un problème d'optimisation avec contrainte.

   Ce poblème se résout avec des techniques classiques implémentées dans `Scicoslab` et qui utilisent vos cours d'optimisation de 1A (passé) et de 2A (futur !).

   La fonction `[f,xopt]=optim(cost,x0)` minimise une fonction, si on lui fournit une fonction `cost` qui renvoie la valeur du coût ainsi que de la dérivée du coût en fonction des paramètres. `x0` est la valeur initiale de l'algorithme. `optim` renvoie la valeur de l'optimum dans `f` et le minimiseur dans `xopt`.

```
// Choix des caracteristiques des actifs sans risque
d=30;rho=0.0;
min_esp=0.05;max_esp=0.15;
mu = [min_esp:(max_esp-min_esp)/(d-1):max_esp];
min_sigma=0.1;max_sigma=0.3;
sigma = [min_sigma:(max_sigma-min_sigma)/(d-1):max_sigma];
correlation = rho*ones(d,d)+(1-rho)*eye(d,d);
Gamma = diag(sigma) * correlation * diag(sigma);

// Définition de la fonction de cout et de sa dérivée
// en utilisant la contrainte sum_{i=1}^d lambda_i = 1
function [f,g,ind]=cost(x,ind)
// On maximise
//       f = (mu*lambda)^2 / lambda'*Gamma*lambda
// sous la contrainte sum_{i=1}^d lambda_i = 1, x = lambda(2:d)
  p=prod(size(x))+1;// dimension de lambda = 1 + dim(x)
  // x_1 est calculé en fonction de lambda à partir de x(2:d)
  // en utilisant la contrainte sum_{i=1}^d lambda_i = 1
  lambda=[1-sum(x);x];
  ps=mu*lambda;
  var=lambda'*Gamma*lambda;
  // On cherche à minimiser (lambda.mu)^2 / lambda'.Gamma.lambda
  f=ps^2 / var;
  // le dérivée du coût en fonction de lambda
  k=(2*ps/var)*mu - (2*ps^2/var^2)*lambda'*Gamma;
  // Calcul de la dérivée par rapport a x
  // en fonction de la derivee en lambda.
  g=k(2:p)-k(1);
  // On maximise mais Scicoslab suppose que l'on minimise ...
  f=-f;
  g=-g;
endfunction

x0=ones(d-1,1)/d;
[f,xopt]=optim(cost,x0);
Xopt=[1-sum(xopt);xopt];
Fopt=sqrt(-f)

// Est on bien entre 0 et 1 ? C'est toujours le cas pour
// Rho diagonale mais pas toujours dans le cas non diagonal
ok = and(0<=Xopt) & and(Xopt<=1)

// Lorsque rho=0, il y a une solution explicite (exercice)
// lambda_i = alpha * mu_i/sigma_i^2, renormalisé
// On verifie ...
```

```
sigma=sqrt(diag(Gamma))';
x=(mu ./ sigma^2);
x=x'/sum(x);// noramlisation
norm(x - Xopt) // lorsque la matrice Rho est diagonale
               // ca devrait etre petit
```

2. La frontière efficiente peut se calculer en résolvant une famille de problème d'optimisation classique : minimisation de variance à espérance fixée et/ou maximisation d'espérance à variance fixée.

```
function lambda=x2lambda(x,R,mu)
  // construction de lambda a partir de x
  // en tenant compte des contraintes sum_i lambda_i = 1, r^T lambda = R
  p=prod(size(x))+2;// dimension de lambda = 2+dim x
  alpha0=1-sum(x);
  beta0=R-mu(1:p-2)*x;
  lambda_n_1 = (mu(d)*alpha0 - beta0) / (mu(d) - mu(d-1));
  lambda_n = (beta0 - mu(d-1)*alpha0) / (mu(d) - mu(d-1));
  lambda=[x;lambda_n_1;lambda_n];
endfunction

function [f,g,ind]=cost(x,ind)
// On minimise la variance
//       f = lambda'*Gamma*lambda
// sous les contraintes sum(lambda) = 1, r^T lambda = R
  p=prod(size(x))+2;// dimension de lambda = 2+dim x
  lambda=x2lambda(x,R,mu);
  k=Gamma*lambda;// derive en fonction de lambda
  f=lambda'*k;// = lambda'*Gamma*lambda;
  for i=[1:p-2] do
    // derivee par rapport a x (et non lambda)
    g(i)=k(i) ...
      + (k(p-1) * (- mu(p) + mu(i)) + k(p) * (- mu(i) + mu(p-1))) ...
          / (mu(d) - mu(d-1));
  end
endfunction

// Choix des caracteristiques des actifs sans risque
d=30;rho=0.0;
min_esp=0.05;max_esp=0.15;
mu=[min_esp:(max_esp-min_esp)/(d-1):max_esp];
min_sigma=0.1;max_sigma=0.3;
sigma=[min_sigma:(max_sigma-min_sigma)/(d-1):max_sigma];
correlation = rho*ones(d,d)+(1-rho)*eye(d,d);
Gamma = diag(sigma) * correlation * diag(sigma);

x0=ones(d-2,1)/d;
i=0;abscisse=0;ordonnee=0;
for R=[0.05:0.001:0.15] do
  [f,xopt]=optim(cost,x0);
  Xopt=x2lambda(xopt,R,mu);
  i=i+1;
  abscisse(i)=sqrt(f);
  ordonnee(i)=R;
end;
plot2d(abscisse,ordonnee);
```

3. Toujours en ignorant les contraintes de positivités, on peut obtenir une forme quasi explicite pour la frontière de Pareto, en utilisant deux multiplicateurs de Lagrange (l'un pour $\sum_{i=1}^{d} \lambda_i = 1$, l'autre pour $\sum_{i=1}^{d} \mu_i \lambda_i = r$).
On obtient après quelques calculs simples, si

$$A = \mu' \Sigma^{-1} \mu,\ B = \mu' \Sigma^{-1} \mathbf{1},\ C = \mathbf{1}' \Sigma^{-1} \mathbf{1},\ D = AC - B^2,$$

la paramètrisation suivante de la frontière de Pareto : $(\lambda' \mu, \lambda' \Sigma \lambda)$ où λ est la fonction de r suivante :

$$\lambda = \frac{BC}{D} \left(r - \frac{B}{C} \right) (\lambda_\mu - \lambda_g) + \lambda_g,$$

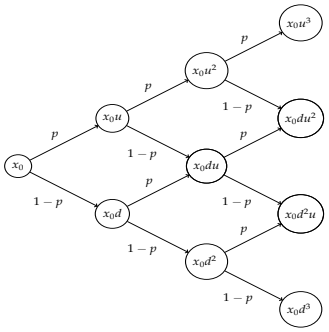
où $\lambda_g = \Sigma^{-1} \mathbf{1} / C$ et $\lambda_\mu = \Sigma^{-1} \mu / B$. On peut vérifier que seule la partie de cette courbe correspondant à $r \geq \frac{B}{C}$ appartient à la frontière de Pareto.
On pourra se convaincre que λ_g est le point qui *minimise* la variance des portefeuilles sous la seule contrainte que $\sum_{i=1}^{d} \lambda_i = 1$ et que λ_μ est le point qui *maximise* le ratio de Sharpe $\mu' \lambda / \sqrt{\lambda' \Sigma \lambda}$ sous la même contrainte (utiliser Cauchy-Schwartz pour s'en convaincre).
Voir T.J. Brennan and A.W. Lo, 2010, "Impossible frontiers" ou Merton, R., 1972, "An analytic derivation of the Efficient Portfolio Frontier".

Loi et chaîne de Markov

Test d'équirépartition, Calcul de prix

Bernard Lapeyre

http://cermics.enpc.fr/~bl



Plan

Probabilité et espérance conditionnelle

- Une *probabilité* $\mathbb{P}$ sur $(\Omega, \mathcal{A})$.
- Une espérance $\mathbb{E}$ (qui opére *linéairement* sur des variables aléatoires à valeurs dans $\mathbb{R}$).
- *Probabilité conditionnelle* : $A, B \subset E, \mathbb{P}(B) > 0$ par définition

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

- *Espérance conditionnelle* : $B \subset E, \mathbb{P}(B) > 0$, X v.a., par définition

$$\mathbb{E}(X|B) = \frac{\mathbb{E}(X\mathbf{1}_B)}{\mathbb{P}(B)}$$

- $\mathbb{E}(\mathbf{1}_A|B) = \mathbb{P}(A|B)$, linéarité.

Loi d'un variable aléatoire

- Un espace d'état fini $E = \{x_1, \ldots, x_n, \ldots\}$, une variable aléatoire X à valeur dans $\mathbb{E}$. Loi de X

$$\mathbb{P}(X = x) = p(x), x \in E$$

- $\sum_{x \in E} p(x) = 1$.
- A quoi ca sert ? A calculer $\mathbb{E}(f(X))$ pour toute fonction f de E dans $\mathbb{R}$!

$$\mathbb{E}(f(X)) = \sum_{x \in E} p(x) f(x).$$

Loi d'un vecteur aléatoire I

- $X = (X_1, \dots, X_n)$ un vecteur aléatoire valeur dans E^n. Loi de X

$$p_X(x_1, \dots, x_n) = \mathbb{P}\left(X_1 = x_1, \dots, X_n = x_n\right), x_1, \dots, x_n \in E$$

- $\sum_{x_1,\dots,x_n \in E} p_X(x_1, \dots, x_n) = 1$.
- A quoi ca sert? A calculer $\mathbb{E}(f(X_1, \dots, X_n))$ pour toute fonction f de E^n dans $\mathbb{R}$!

$$\mathbb{E}(f(X_1, \dots, X_n)) = \sum_{x_1,\dots,x_n \in E} p_X(x_1, \dots, x_n) f(x_1, \dots, x_n).$$

- Par *contraction*, la loi de X permet de calculer les lois p^i des X_i

$$p_{X_i}(x) = \sum_{x_1,\dots,x_{i-1},x_{i+1},\dots,x_n \in E} p_X(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n).$$

Loi d'un vecteur aléatoire II

- Une remarque simple (mais utile) :

$$\mathbb{E}(f(X_n)) = \sum_{x_n \in E} p_{X_n}(x_n) f(x_n) = \sum_{x_1,\dots,x_n \in E} p_X(x_1, \dots, x_n) f(x_n).$$

LOI D'UN VECTEUR ET INDÉPENDANCE

- $(X_1, \ldots, X_n)$ est un vecteur de v.a. indépendantes si et seulement si (au choix), pour tout $x_1, \ldots, x_n \in E$, pour tout $f_1, \ldots, f_n$ de E dans $\mathbb{R}$:
  - $\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \mathbb{P}(X_1 = x_1) \times \ldots \times \mathbb{P}(X_n = x_n)$
  - $\mathbb{E}(f_1(X_1) \times \cdots \times f_n(X_n)) = \mathbb{E}(f_1(X_1)) \times \cdots \times \mathbb{E}(f_n(X_n))$
- les lois marginales p^i des X_i *ne caractérisent pas* la loi du vecteur $(X_1, \ldots, X_n)$!
- ... sauf si on suppose que ces v.a. sont *indépendantes*.

SYSTÈME DYNAMIQUE ALÉATOIRE

- F application de E dans E permet de définir un *système dynamique*: x_0 arbitraire, puis

$$x_{n+1} = F(x_n).$$

- $W = (W_n, n \geq 1)$ suite de variables aléatoires indépendantes de même loi sur un espace G. "On tire une application de E dans E, au hasard" : $F(x, W_{n+1})$.
- Sytème dynamique aléatoire, x_0 arbitraire, on définit par récurrence

$$X_{n+1} = F(X_n, W_{n+1})$$

- La connaissance de $F(x, w)$ permet de simuler très simplement la suite $(X_n, n \geq 0)$ à partir de la suite W.

SYSTÈMES DYNAMIQUES ALÉATOIRES : 3 EXEMPLES

$(W_n, n \geq 1)$ une suite de tirages à pile ou face indépendants, $0 \leq p \leq 1$.

$$\mathbb{P}(W_n = P) = p = 1 - \mathbb{P}(W_n = F)$$

- **Marche aléatoire** ($p = 1/2$, marche aléatoire symétrique)

$$X_0 = 0, X_{n+1} = X_n + \left(\mathbf{1}_{\{W_{n+1} = P\}} - \mathbf{1}_{\{W_{n+1} = F\}}\right).$$

- **Processus en "dents de scie"**: (# pile consécutifs avant n)

$$X_0 = 0, X_{n+1} = (1 + X_n)\mathbf{1}_{\{W_{n+1} = P\}}.$$

- **Processus de Cox-Ross** (cf. finance), $d < 1 < u$

$$X_0 = 1, X_{n+1} = X_n \left(u\, \mathbf{1}_{\{W_{n+1} = P\}} + d\, \mathbf{1}_{\{W_{n+1} = F\}}\right).$$

- autres exemples : gestion de stock, battage de cartes (voir [Berestycki(2007)]), ...

Système dynamique aléatoire et propriété de Markov

- Un système dynamique aléatoire vérifie la *propriété de Markov*

$$\mathbb{P}\left(X_{n+1}=x_{n+1}|X_0=x_0,\dots,X_{n-1}=x_{n-1},X_n=x_n\right)=P(x_n,x_{n+1})$$

- où P une matrice de transition : $(P(x,y),x,y\in E)$ (parfois aussi noté $(P(x\to y))$ donnée par

$$P(x,y)=\mathbb{P}(F(x,W_1)=y).$$

- *Matrice de transition* : $P(x,y)\geq 0$ et $\sum_{y\in E}P(x,y)=1$.
- On a forcement $P(x,y)=\mathbb{P}\left(X_{n+1}=y|X_n=x\right)$ et

$$\mathbb{P}\left(X_{n+1}=x_{n+1}|X_0=x_0,\dots,X_{n-1}=x_{n-1},X_n=x_n\right)=\mathbb{P}\left(X_{n+1}=x_{n+1}|X_n=x_n\right)$$

Preuve

$$\begin{aligned}&\mathbb{P}\left(X_{n+1}=x_{n+1},X_n=x_n,\dots,X_0=x_0\right)\\&\quad=\mathbb{P}\left(F(X_n,W_{n+1})=x_{n+1},X_n=x_n,\dots,X_0=x_0\right)\\&\quad=\mathbb{P}\left(F(x_n,W_{n+1})=x_{n+1},X_n=x_n,\dots,X_0=x_0\right).\end{aligned}$$

- $F(x_n,W_{n+1})$ indépendante du vecteur $(W_1,\dots,W_n)$
- $X_0,\dots,X_n$ fonctions de $(W_1,\dots,W_n)$.
- Donc $F(x_n,W_{n+1})$ indépendante de $X_0,\dots,X_n$.

$$\begin{aligned}&\mathbb{P}\left(X_{n+1}=x_{n+1},X_n=x_n,\dots,X_0=x_0\right)\\&\quad=\mathbb{P}\left(F(x_n,W_{n+1})=x_{n+1}\right)\mathbb{P}\left(X_n=x_n,\dots,X_0=x_0\right)\\&\quad=P(x_n,x_{n+1})\mathbb{P}\left(X_n=x_n,\dots,X_0=x_0\right)\end{aligned}$$

- En sommant sur tous les $x_0,\dots,x_{n-1}$ dans E, on obtient aussi

$$\mathbb{P}\left(X_{n+1}=x_{n+1},X_n=x_n\right)=P(x_n,x_{n+1})\mathbb{P}\left(X_n=x_n\right).$$

MATRICE DE TRANSITION : EXEMPLES

- Exemple 1 : $x \in \mathbf{Z}$, $P(x, x+1) = p$, $P(x, x-1) = 1-p$, 0 sinon
- Exemple 2 : $x \in \mathbb{N}$, $P(x, x+1) = p$, $P(x, 0) = 1-p$, 0 sinon
- Exemple 3 : $P(x, xu) = p$, $P(x, xd) = 1-p$, 0 sinon

SYSTÈME DYNAMIQUE ALÉATOIRE ET CHAÎNE DE MARKOV

- une suite de variables aléatoires qui vérifie la propriété de Markov, s'appelle une *chaîne de Markov*.
- $P(x, y) = \mathbb{P}(X_{n+1} = y | X_n = x)$, $x, y \in E$, est la *matrice de transition* de la chaîne de Markov.
- Un système dynamique aléatoire est une chaîne de Markov et reciproquement on peut représenter (en loi) une chaîne de Markov comme un système dynamique aléatoire.
- La matrice de transition $P(x, y)$ se déduit de F et de la loi de W.
- P contient toute l'information utile pour mener à bien les calculs de loi concernant la chaîne.
- Pour *simuler* une chaîne de Markov, on utilise une représentation comme système dynamique aléatoire.

Chaîne de Markov : définition

Definition 1

Une suite de v.a. $(X_n, n \geq 0)$ à valeurs dans E est un chaîne de Markov de *matrice de transition* $(P(x,y), x \in E, y \in E)$, si et seulement si, par définition, pour tout $x_0, x_1, \ldots, x_n, x_{n+1} \in E$,

$$\begin{aligned}\mathbb{P}\left(X_{n+1} = x_{n+1} | X_0 = x_0, \ldots, X_{n-1} = x_{n-1}, X_n = x_n\right) \\ = P(x_n, x_{n+1}) \\ = \mathbb{P}\left(X_{n+1} = x_{n+1} | X_n = x_n\right)\end{aligned}$$

La loi de X_0, μ_0, est appelée la *loi initiale*.

- Formellement, égalité vrai seulement si $\mathbb{P}(X_0 = x_0, \ldots, X_{n-1} = x_{n-1}, X_n = x_n) > 0$, sinon l'interpréter comme

$$\begin{aligned}\mathbb{P}(X_0 = x_0, \ldots, X_{n-1} = x_{n-1}, X_n = x_n, X_{n+1} = x_{n+1}) \\ = \mathbb{P}(X_0 = x_0, \ldots, X_{n-1} = x_{n-1}, X_n = x_n) P(x_n, x_{n+1}).\end{aligned}$$

- Matrice de transition P_n dépendant de n : chaîne de Markov *inhomogène*.

MATRICE DE TRANSITION ET CALCUL DE LOIS

- E fini
- $(X_n, n \geq 0)$ une chaîne de Markov de matrice de transition P sur E.
- μ_0 la loi de X_0 (loi initiale), $\mu_0(x) = \mathbb{P}(X_0 = x)$, $x \in E$.
- La loi du vecteur $(X_0, \ldots, X_n)$ se calcule explicitement en fonction de P et μ_0

$$\mathbb{P}(X_0 = x_0, \ldots, X_n = x_n) = \mu_0(x_0) P(x_0, x_1) \times \cdots \times P(x_{n-1}, x_n).$$

- Preuve = propriété de Markov

$$\begin{aligned}\mathbb{P}(X_0 = x_0, \ldots, X_{n-1} = x_{n-1}, X_n = x_n) \\ = \mathbb{P}(X_0 = x_0, \ldots, X_{n-1} = x_{n-1})\, P(x_{n-1}, x_n),\end{aligned}$$

puis on itère.

MATRICE DE TRANSITION : QUELQUES NOTATIONS

- E fini : μ est un vecteur *ligne*, P une *matrice* et f un vecteur *colonne*.
- $(Pf)(x) = \sum_{y \in E} P(x,y) f(y)$, f une fonction de E dans $\mathbb{R}$. Pf est aussi une fonction de E dans $\mathbb{R}$.
- $(\mu P)(y) = \sum_{x \in E} \mu(x) P(x,y)$, μ une loi de probabilité sur E. μP est aussi une probabilité sur E
- $(PQ)(x,y) = \sum_{z \in E} P(x,z) Q(z,y)$, pour P et Q deux matrices de transition. PQ est aussi une matrice de transition.
- P^n est définie par récurrence par :

$$P^{n+1}(x,y) = \sum_{z \in E} P(x,z) P^n(z,y).$$

- μ loi sur E, f fonction de E dans $\mathbb{R}$: $\mu f = \sum_{x \in E} \mu(x) f(x)$.
- Si X a pour loi μ, $\mathbb{E}(f(X)) = \mu f$.

Loi de X_n

- Par contraction de la loi du vecteur $(X_0, \dots, X_n)$, on obtient

$$\mathbb{P}(X_n = x_n) = \sum_{x_0, x_1, \dots, x_{n-1} \in E} \mu_0(x_0) P(x_0, x_1) \times \cdots \times P(x_{n-1}, x_n).$$

- avec les notations précédentes :

$$\mathbb{P}(X_n = x_n) = (\mu_0 P^n)(x_n),$$

- $\text{loi}(X_n) = \mu_0 P^n$: un (simple) calcul matriciel à partir de μ_0 et P.

UN TEST D'ÉQUIRÉPARTITION

- Ces 100 tirages à pile ou face (réalisé par mes soins!) sont ils vraiment "aléatoire"?

P	F	F	F	P	F	P	P	F	F
F	P	P	P	F	F	P	F	P	F
P	F	F	F	P	P	F	F	P	P
P	F	F	P	F	P	P	F	F	P
F	P	P	F	P	P	F	F	F	P
P	P	F	P	F	P	P	F	P	P
F	F	P	F	P	P	F	F	P	F
P	F	F	F	P	F	P	P	F	F
P	F	F	F	P	P	P	F	P	F
P	F	F	P	P	F	P	F	P	P

- Ça n'a pas l'air trop mal : 50P, 50F ...

UN TEST D'ÉQUIRÉPARTITION

- mais ... le nombre maximum de piles consécutifs est de 3, ce qui est (très) peu compatible avec l'hypothèse d'un tirage au hasard.
- On a du mal a imiter le hasard, ...
- Nous allons voir qu'il faut s'attendre à 4 piles consécutifs (avec proba 0.97) voire 5 (avec proba 0.81).
- On a plus de chance de voir 10 piles consécutifs (ou plus) que d'en voir 3 ou moins !
- On va calculer le loi du nombre maximum de piles consécutifs au bout de 100 tirages.

Chaîne de Markov arrêtée

- $(X_n, n \geq 0)$ un système dynamique aléatoire

$$X_{n+1} = F(X_n, W_{n+1}).$$

- $x \in E$, $\tau = \inf\{n \geq 0, X_n = x_0\}$: le premier temps d'atteinte de x_0 (pas forcément fini, $+\infty$ si ensemble vide).
- $n \wedge \tau = \inf\{n, \tau\}$
- $Y_n = X_{n\wedge\tau}$ est aussi un système dynamique aléatoire (et donc une chaîne de Markov) puisque

$$Y_{n+1} = F(Y_n, W_{n+1})\mathbf{1}_{\{Y_n \neq x_0\}} + x_0\mathbf{1}_{\{Y_n = x_0\}} = G(Y_n, W_{n+1}).$$

$$G(x,u) = F(x,u)\mathbf{1}_{\{x \neq x_0\}} + x_0\mathbf{1}_{\{x = x_0\}}$$

Chaîne en dents de scie arrêtée

- X la chaîne en dents de scie, $l > 0$, $\tau_l = \inf\{n \geq 0, X_n = l\}$, $Y^l_n = X_{n\wedge\tau_l}$.
- Y^l_n est un chaîne de Markov à valeurs dans $\{0, \ldots, l\}$ de matrice de transition P donnée par (les autres élèments sont nuls) :

$$\begin{cases} P_l(x, x+1) = p & \text{si } x < l, \\ P_l(x, 0) = 1-p & \text{si } x < l, \\ P_l(l,l) = 1. \end{cases}$$

$$P_l = \begin{pmatrix} 1-p & p & 0 & \ldots & 0 & 0 \\ 1-p & 0 & p & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1-p & 0 & 0 & \ldots & p & 0 \\ 1-p & 0 & 0 & \ldots & 0 & p \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix}.$$

Loi du nombre de piles consécutifs après 100 tirages

- $N^n_{\max}$: nombre maximum de piles consécutifs après n tirages.
- $\{Y^l_n = l\} = \{N^n_{\max} \geq l\}$.
- $\mathbb{P}(Y^l_0 = 0) = 1$. Loi initiale $\mu_0(0) = 1$, $\mu_0(k) = 0$ sinon. $\mu_0 = (1, 0, \ldots, 0)$.
- Loi de $Y^l_n = \mu_0(P_l)^n = (P^n_l(0,k), 0 \leq k \leq l)$ (un vecteur ligne).
- $\mathbb{P}(Y^l_n = l) = (P_l)^n(0,l)$.
- Il est facile d'écrire un programme`Python` qui fait cela (cf. TD).
- On peut calculer la *loi* du nombre de piles consécutifs maximum après 100 tirages $N_{\max}$.

$$\begin{aligned}\mathbb{P}(N^{100}_{\max} = l) &= \mathbb{P}(Y^l_{100} = l) - \mathbb{P}(Y^{l+1}_{100} = l+1) \\ &= P^{100}_l(0,l) - P^{100}_{l+1}(0,l+1).\end{aligned}$$

Loi du nombre maximum de piles consécutifs

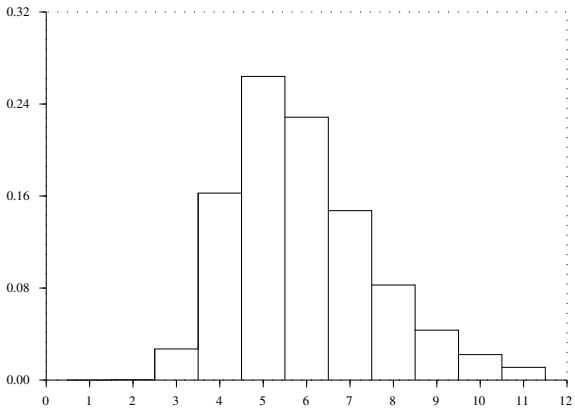


Figure: Loi du nombre de piles consécutifs.

- $\mathbb{P}(\text{nombre de piles consécutifs } \leq 3) = 0.0273$
- $\mathbb{P}(\text{nombre de piles consécutifs } \geq 5) = 0.8101$

UNE RÉGLE DE DÉCISION

- Si l'on voit au moins 4 piles consécutifs, la suite sera déclarée aléatoire, sinon, la suite sera déclarée non aléatoire.
- Si la suite a été tirée aléatoirement, je vais me tromper avec moins de 3% d'erreur.
- Si la suite n'est pas aléatoire, je suis aidé par la tendance naturelle à ne pas faire de trop longues series.
- "Mon" exemple ne passe pas le test. Mais si on le lit par colonne ou lieu de ligne, il le passe aisément!
- À vérifier avec l'un de vos camarades, si vous êtes sceptique.

LOI DE X_n : UNE AUTRE FAÇON D'ÉCRIRE LES CHOSES

- $(X_n, n \geq 0)$ chaîne de Markov de matrice de transition P sur E.
- $\mathbb{P}(X_0 = x_0) = 1$ (i.e. $\mu_0(x_0) = 1$, $\mu_0(x) = 0$ si $x \neq x_0$.)
- On sait exprimer $\mathbb{E}(f(X_N))$

$$\mathbb{E}(f(X_N)) = \mu_N f = \mu_0 P^N f = \sum_{x \in E} \mu_0(x)(P^N f)(x) = (P^N f)(x_0).$$

Une formulation alternative plus algorithmique.

Theorem 2

Soit $(u(n,x), n = 0, \dots, N, x \in E)$ la solution unique de

$$(1) \qquad \begin{cases} u(n,x) = \sum_{y \in E} P(x,y) u(n+1,y), & n < N \\ u(N,x) = f(x), \end{cases}$$

Alors $\mathbb{E}(f(X_N)) = u(0, x_0)$.

REMARQUES

- La première équation de (1) peut aussi s'écrire

$$u(n,x) = P[u(n+1,.)](x) = \sum_{y \in E} P(x,y) u(n+1,y)$$

- $u(n,x)$ peut s'interpréter comme

$$u(n,x) = \overbrace{P \times \cdots \times P}^{N-n \text{ fois}} f(x) = P^{N-n} f(x),$$

- Lorsque $\mathbb{P}(X_n = x) > 0$, on a

$$u(n,x) = \mathbb{E}\left(f(X_N) | X_n = x\right) = P^{N-n} f(x).$$

PREUVE

$$\begin{aligned}&\mathbb{P}(X_0 = x_0, \dots, X_n = x_n, X_{n+1} = x_{n+1}, \dots, X_N = x_N)\\ &\quad = \mathbb{P}(X_0 = x_0) P(x_0, x_1) \times \cdots \times P(x_n, x_{n+1}) \times P(x_{n+1}, x_{n+2}) \times \cdots \times P(x_{N-1}, x_N)\end{aligned}$$

en sommant sur toutes les valeurs de $x_0, \cdots, x_{n-1}$ et $x_{n+1}, \cdots, x_{N-1}$ on obtient la loi de (X_n, X_N)

$$\mathbb{P}(X_n = x_n, X_N = x_N) = \mathbb{P}(X_n = x_n) P^{N-n}(x_n, x_N)$$

$$\begin{aligned}\mathbb{E}\left(f(X_N)\mathbf{1}_{\{X_n = x_n\}}\right) &= \mathbb{P}(X_n = x_n) \sum_{x_N \in E} P^{N-n}(x_n, x_N) f(x_N)\\ &= \mathbb{P}(X_n = x_n)(P^{N-n} f)(x_n)\end{aligned}$$

REMARQUES ET NOTATIONS

- (1) est *une équation de programmation dynamique*
- E est fini, (1) est un *algorithme* qui termine.

UNE NOTATION COMMODE "À LA `Scilab`"

- $x_{0:n}$ plutôt que $(x_0, \dots, x_n)$
- $X_{0:n}$ plutôt que $(X_0, \dots, X_n)$
- $X_{0:n} = x_{0:n}$ plutôt que $(X_0 = x_0, \dots, X_n = x_n)$

PREUVE FORMELLE

On le sait déjà ... mais voici une autre méthode de preuve. On va montrer que

$$\mathbb{E}\left(u(n+1, X_{n+1})\right) = \mathbb{E}\left(u(n, X_n)\right).$$

Si cela est vrai :

$$u(0, x_0) = \mathbb{E}\left(u(0, X_0)\right) = \cdots = \mathbb{E}\left(u(N, X_N)\right) = \mathbb{E}\left(f(X_N)\right).$$

La loi de $X_{0:n+1} = (X_{0:n}, X_{n+1})$ est donnée par (c'est un façon d'exprimer la propriété de Markov)

$$\mathbb{P}\left(X_{0:n+1} = x_{0:n+1}\right) = \mathbb{P}\left(X_{0:n} = x_{0:n}, X_{n+1} = x_{n+1}\right) = \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}).$$

$$\begin{aligned}
\mathbb{E}\left(u(n+1, X_{n+1})\right) &= \sum_{x_{0:n+1} \in E^{n+2}} u(n+1, x_{n+1}) \mathbb{P}\left(X_{0:n+1} = x_{0:n+1}\right), \\
\text{[propriété de Markov]} &= \sum_{x_{0:n} \in E^{n+1}, x_{n+1} \in E} u(n+1, x_{n+1}) \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}), \\
&= \sum_{x_{0:n} \in E^{n+1}} \mathbb{P}\left(X_{0:n} = x_{0:n}\right) \sum_{x_{n+1} \in E} P(x_n, x_{n+1}) u(n+1, x_{n+1}), \\
\text{[définition de } u(n,\,.)\text{]} &= \sum_{x_{0:n} \in E^{n+1}} \mathbb{P}\left(X_{0:n} = x_{0:n}\right) u(n, x_n) = \mathbb{E}\left(u(n, X_n)\right).
\end{aligned}$$

CALCUL DU PRIX D'UNE OPTION

- Un modèle : le processus de Cox-Ross

$$X_0 = x_0, X_{n+1} = X_n \left(u\, \mathbf{1}_{\{W_{n+1} = P\}} + d\, \mathbf{1}_{\{W_{n+1} = F\}}\right).$$

- Un profil d'option $f(X_N)$ (ce que je vais recevoir en N)

$$f(x) = (x - K)_+, \text{ option d'achat de prix d'exercice } K.$$

- Prix : $\mathbb{E}(f(X_N))$ ("intuitif", voir cours de maths financière 2A).
- Calcul du prix : $(u(n,x), n = 0, \ldots, N, x \in E)$

$$(2) \qquad \begin{cases} u(n,x) = pu(n+1, xu) + (1-p)u(n+1, xd), \\ u(N,x) = f(x). \end{cases}$$

- $\mathbb{E}(f(X_N)) = u(0, x_0)$.

UN PROGRAMME

Un programme en `Python`, catastrophique (complexité 2^N), mais qui a le bon goût de terminer (lentement)

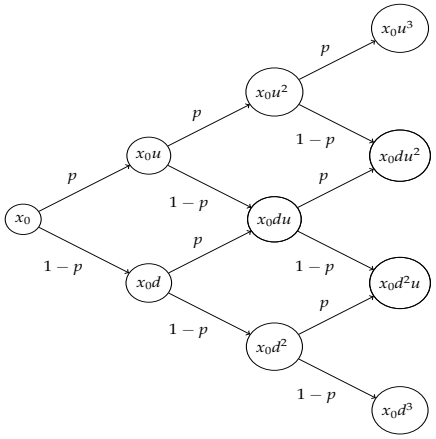
```
N=10;p=1/2;
down=1/2;up=2;
K=1;

def f(x): return max(x-K,0)

def u(n,x):
    if n==N:
        return f(x);
    else:
        return p * u(n+1,x*up)+ (1-p) * u(n+1,x*down);

def prix_rec(x): return u(0,x)
```


POUR ALLER PLUS VITE, UN DESSIN POUR $N = 3$



UN ALGORITHME QUI FAIT LA MÊME CHOSE, PLUS VITE ...

- Si $X_0 = x_0$ les seules valeurs que peut prendre X_n sont données par (k = nombre de fois où l'on tire u entre 0 et $n-1$)

$$(x_k^{(n)} = x_0 u^k d^{n-k}, 0 \leq k \leq n).$$

- Il suffit de calculer les vecteurs $U^{(n)} = (u(n, x_k^{(n)}), 0 \leq k \leq n)$, qui vérifient

$$(3) \qquad \begin{cases} U^{(N)}(k) = f(x_k^{(N)}), & k = 0, \dots, N \\ U^{(n)}(k) = pU^{(n+1)}(k+1) + (1-p)U^{(n+1)}(k), & k = 0, \dots, n. \end{cases}$$

- Ce qui donne un agorithme (de complexité N^2).

UN ALGORITHME QUI FAIT LA MÊME CHOSE, PLUS VITE ...

```
def prix_iter(x_0):
  U=np.zeros((N+1,N+1))
  for k in range(N+1): # = intervalle [0,1,...,N]
    U[N,k] = f(x_0 * up**k * down**(N-k))

  for n in range(N-1,-1,-1):  # = intervalle [N-1,N-2,...,0]
    for k in range(n+1):  # intervalle [0,1,...,n]
        U[n,k] = p*U[n+1,k+1]+(1-p)*U[n+1,k]
  return U[0,0]
```

ANDRY MARKOV

- Mathematicien russe 1856 – 1922, élève de Chebyshev, "known for his work in number theory, analysis, and probability theory."
- ... "He introduced a new sequence of dependent variable, called a chain, as well as a few basic concepts of chains such as transition probabilities, irreductibility and stationarity. His ideas were taken up and developed further by scientists around the world and now the theory of Markov Chains is one of the most powerful theories for analyzing various phenomena of the world"

A. A. Марков (1886).

Pour des détail sur la vie et l'œuvre de Markov voir [Basharin et al.(2004)Basharin, Langville, and Naumov].

POUR ALLER PLUS LOIN : COURS DE L'ECOLE DES PONTS

- 2A : cours "Processus Aléatoires"
- cours "Méthodes Mathématiques pour la Finance"

BIBLIOGRAPHIE


Gely P. Basharin, Amy N. Langville, and Valeriy A. Naumov.
The life and work of A. A. Markov.
Linear Algebra Appl., 386:3–26, 2004.


Nathanaël Berestycki.
Notes on card shuffling.
Technical report, http://www.statslab.cam.ac.uk/~beresty/Articles/shuffle.pdf, 2007.


Jean-François Delmas and Benjamin Jourdain.
Modèles aléatoires.
Mathématiques & Applications. Springer-Verlag, Berlin, 2006.


Stuart Dreyfus.
Richard bellman on the birth of dynamic programming.
Operations Resarch, 50(1):48–51, 2002.


Carl Graham.
Chaînes de Markov.
Dunod, 2008.


Damien Lamberton and Bernard Lapeyre.
Introduction au calcul stochastique appliqué à la finance.
Ellipses, Édition Marketing, Paris, second edition, 1997.

Cours de Décision dans l'incertain
Exercices : mercredi 29 avril 2020.

Exercice 1 On considère une suite de variables aléatoires indépendantes de même loi $(U_n, n \geq 1)$ prenant ces valeurs dans G (noter qu'il n'est pas nécessaire de supposer G fini). Soit E un ensemble fini et F une application de $E \times G$ dans E.

On construit, par récurrence, une suite $(X_n, n \geq 0)$ en posant, $X_0 = x \in E$ et

$$X_{n+1} = F(X_n, U_{n+1}), n \geq 0.$$

1. Montrer que $(X_n, n \geq 0)$ est une chaîne de Markov et exprimer sa matrice de transition P en fonction de F.
2. On définit τ par
$$\tau = \inf \{n \geq 0, X_n = x_0\}.$$
Montrer que $Y_n = X_{n \wedge \tau}$ est aussi une chaîne de Markov. Calculer sa matrice de transition en fonction de celle de X.

Exercice 2
1. Soit $(X_n, n \geq 0)$ une chaîne de Markov de matrice de transition $(P(x, y), x \in \mathbb{E}, y \in E)$. Calculer la loi du vecteur $(X_0, X_1, \ldots, X_n)$ et en déduire la loi de X_n.
2. Montrer que $P^n(x, y) = \mathbb{P}(X_n = y | X_0 = x)$ (en particulier, que si la chaîne part d'un point déterministe x_0 [i.e. $X_0 = x_0$ avec probabilité 1], alors $\mathbb{P}(X_n = y) = P^n(x_0, y)$).
3. Calculer la loi du couple (X_n, X_N), pour $n < N$, et en déduire que
$$\mathbb{E}(f(X_N)|X_n = x) = P^{N-n} f(x),$$
et, en particulier, que si $X_0 = x_0$ avec probabilité 1, $\mathbb{E}(f(X_N)) = P^N f(x_0)$.

Exercice 3 Soit $(X_n, n \geq 0)$ un processus de Markov de matrice de transition P sur un espace fini E. Pour une fonction f de E dans $\mathbb{R}$, on note $P^0 f(x) = f(x)$, puis récurence

$$P^{k+1} f(x) = \sum_{y \in E} P(x, y) P^k f(y).$$

1. Vérifier que P^k est bien la puissance k-ième (au sens du produit des matrices) de P.

On définit $u(n, x)$ par

$$u(n, x) = P^{N-n} f(x), x \in E, n \leq N,$$

2. Vérifier que l'on a
$$\begin{cases} u(N, x) &= f(x) \quad x \in E \\ u(n, x) &= \sum_{y \in E} P(x, y) u(n+1, y) \quad x \in E, n < N. \end{cases}$$
3. Montrer que $\mathbb{E}(u(n, X_n)) = \mathbb{E}(u(n+1, X_{n+1}))$ pour $0 \leq n < N$.
4. Redémontrer à partir de cette propriété, le résultat de l'exercice précédent : si la chaîne part de x_0 à l'instant 0, $\mathbb{E}(f(X_n)) = u(0, x_0) = P^N f(x_0)$.

Chaîne de Markov : calcul de loi

Bernard Lapeyre
CERMICS, École des Ponts ParisTech

3 mars 2020

Table des matières

Les programmes suivant on été prévu pour être exécutés à l'aide `Scicoslab`. Ce programme peut être téléchargé librement sur `www.scicoslab.org`. Il est disponible sur MacOSX, Windows ou Linux (Debian, Ubuntu ou Fedora). Noter toutefois qu'il faut installer le server `X`, `XQuartz` sur MacOSX. Ces programmes peuvent aussi fonctionner, moyennant de légère modification, avec la version officielle de `Scilab`.

Le fichier source en `Scilab` correspondant à ce document est accessible. Il est partiellement mais pas totalement corrigé. Le correction complète sera accessible à la fin du TD.

1 Un test d'équirépartition

On commence par construire un test d'équirpartition d'une suite de 100 tirages à pile ou face.

Etude par simulation

On commence par simuler des tirages à pile ou face répétés. On utilise la fonction `scilab`, `grand` avec l'option `"bin"` (loi binomiale) mais avec un seul tirage (donc loi de Bernouilli). A partir de ce tirage on calcule le nombre de pile consécutifs maximum dans le vecteur.

1. Ecrire une fonction qui génére 100 tirages à pile ou face équilibrés ($p = 1/2$, tirages indépendants) et une fonction qui calcule le nombre maximum de P consécutifs sur ces tirages.

```
function X=tirage_pf(n,p)
// Effectue n tirage a Pile (P) ou face (F) (p,1-p)
  X=grand(1,n,"bin",1,p);
  X=string(X);// transforme le tableau de nombres en
              // tableau de caractères
  X=strsubst(X,'0','F');// remplace les 0 par des F
  X=strsubst(X,'1','P');// remplace les 1 par des P
endfunction
```

```
function [MAX] = max_length(U)
// Calcule le nombre maximum de 1 consecutifs
// dans la suite U
  MAX=0;N=0;
  for n=[1:size(U,'*')] do
    QUESTION: if U(n)=='P' then <À COMPLÉTER> else <À COMPLÉTER> end;
    MAX=max(MAX,N);
  end
endfunction
```

2. On aura besoin de tracer des histogrammes.

```
function H=histo_discret(samples, maxsize, varargin)
// histogramme de tirages selon une loi discrète à valeurs entières
// supposé prendre des valeurs entre 0 et max.
// Si tracer_histo=%f pas de dessin
  H=0
  for k=0:maxsize
    // Calcul du nbre de tirages valant k / Taille
    H(k+1)=length(find(samples==k)) ./ size(samples,'*');
  end;

  [lhs,rhs]=argn(0);
  if (rhs == 3) & (varargin(1)==[%t]) then
    xbasc;plot2d3(0:maxsize,H);
    f=gcf();
    Dessin=f.children(1).children(1).children(1);
    Dessin.thickness=10;Dessin.foreground=5;
  end;
endfunction
```

3. Faire 1000 tirages du nombre maximum de P et en tracer un histogramme. On calcule par simulation une approximation de la loi du nombre maximum de piles consecutifs (un histogramme d'un grand nombre de tirages i.i.d.).

```
function main_1()
  p=1/2;
  // On teste cette fonction avec N=20
  // rand(1:20) renvoie 100 tirages uniformes sur l'intervalle [0,1]
  U=tirage_pf(20,p)// 20 tirages a pile/face 1/2,1/2 (1=Pile,0=Face)
  max_length(U)// nombre mximum de P consecutifs

  // On effectue 1000 tirages avec N=100
  N=100;X=0;
  Taille=1000;// nbre de simulation
  for i=[1:Taille] do
    U=tirage_pf(N,p);
    X(i)=max_length(U);
  end;

  // On trace l'histogramme si plot_flag==%t
```

```
  plot_flag=%t
  histo_discret(X,20,plot_flag)
endfunction
```

4. Montrer que le nombre de piles successifs jusqu'à l'instant courant est une chaîne de Markov à valeurs dans $\mathbf{N}$ de matrice de transition $P(x, x+1) = 1/2$, $P(x, 0) = 1/2$. Simuler et tracer de trajectoires de cette chaîne.

```
function [X] = trajectoire(U)
// On calcule la trajectoire de X
  X=0;val=0;
  for n=[1:prod(size(U))] do
    QUESTION: if U(n)=='P' then <À COMPLÉTER> else <À COMPLÉTER> end;
    X(n)=val;
  end
endfunction
```

```
function main_2()
  N=100;
  p=1/2;
  rectangle=[1,0,N,8];

  // On trace une trajectoire de X
  U=tirage_pf(N,p);
  X=trajectoire(U);
  xbasc;plot2d2(1:N,X,rect=rectangle);
  xclick;// attend un click dans la fenêtre graphique

  // 4 autres trajectoires
  for i=1:4 do
    U=tirage_pf(N);
    plot2d2(1:N,trajectoire(U),rect=rectangle,style=i+1);
    xclick;// attend un click dans la fenêtre graphique
  end;
endfunction;
```

Calcul exact de la probabilité

On calcule exactement la probabilité de voir au moins l piles consécutifs.

1. Calculer la matrice de transition de la chaîne arrêté en l, l'implementer en `Scicoslab`. En déduire la probabilité d'avoir au moins l pile consécutifs pour $l = 0, \ldots, 20$.

```
function Pro= proba(N,l,p)
  // Calcule la probabilite de voir au moins l piles consecutifs
  // dans N tirages a pile (p) ou face (1-p)

        // la matrice de transition de la chaîne
        // de taille (l+1,l+1)
        P=[p*diag(ones(l,1));zeros(1,l-1),1];
        P=[[(1-p)*ones(l,1);0],P];
        PN=P^N;
        QUESTION: Pro = QUE VAUT LA PROBA DE VOIR AU MOINS l PILES;
endfunction
```

2. Calculer la loi du nombre maximum de piles consécutifs.

```
function [loi]=calculer_loi(N,p)
  MAXMAX=30;// a choisir assez grand (mais pas trop ...)
  // attention au decalage de 1 pour les vecteurs :
  //      loi(1)=P(X=0), ..., loi(n+1)=P(X=n)

  previous=1;// proba d'avoir au moins 0 pile = 1 !
  // le support de la loi est [0,1,...,N] que l'on tronque en MAXMAX
  for l=0:min(N,MAXMAX) do
    QUESTION: current=<À COMPLÉTER>;// proba d'avoir au moins l+1 pile
    loi(l+1)=previous - current;// proba d'avoir exactement l pile
    previous=current;
  end
  loi=loi';// c'est plus joli comme ca !
endfunction
```

3. Comparer avec les simulations précédentes. Vérifier que $\mathbf{P}(X = 3)$ est du même ordre que $\mathbf{P}(X = 10)$.

```
function main_3()
  // On teste avec N=1 et N=2, p=1/2
  // Pour N=1, 0 pile avec proba 1/2 et 1 pile avec proba 1/2
  calculer_loi(1,1/2)
  // Pour N=2, on doit trouver (1/4,1/2,1/4) pour (0,1,2)
  calculer_loi(2,1/2)
  // en principe ca marche ...

  N=100;p=1/2;
  loi=calculer_loi(N,p);
  sum(loi)// on verifie que ca somme a 1

  // dessin
  // ATTENTION on est decale de 1, donc 0:20 devient 1:21
  xbasc;plot2d3(0:20,loi(1:21));
  f=gcf();Dessin=f.children(1).children(1).children(1);
  Dessin.thickness=10;

  // comparaison avec les simulations
  Taille=10000;
```

```
  for i=[1:Taille] do
    U=tirage_pf(N);
    X(i)=max_length(U);
  end

  histo=0;
  for i=0:20 do
    histo(i+1)=sum(X==i)/Taille;
  end
  histo=histo';

  epsilon=norm(histo(1:21)-loi(1:21))
          // doit etre "petit", pour bien faire il faudrait faire un
          // test du ξ² pour savoir ce que "petit" veut dire ici.
  printf("epsilon=%f, petit en principe.\n",epsilon)
endfunction
```

Test du critère

1. Vérifier que, pour des tirages aléatoires, le test proposé (runs plus grand que 4) fonctionne dans la plupart des cas (97%).

```
function main_4()
// Test du critère lorsque les tirages sont aléatoires
// Ca marche "souvent" mais pas "toujours".
  N=100;
  p=1/2;
  Taille=10;
  for i=[1:Taille] do
    U=tirage_pf(N,p);
    if max_length(U) >= 4 then
      printf("OK\n");
    else
      // Ca arrive "rarement" mais ca arrive 3 fois sur 100
      printf("test négatif\n");
    end
  end;
endfunction;
```

Comment le loi varie t'elle en fonction de N ?

On regarde ce qui se passe lorsque N devient grand.

1. Calculer le maximum de vraisemblance de la loi du "run maximum" en fonction de N ($N = 100$, $N = 500$, $N = 1000$).

```
function main_5()
// Calcul du maximum de vraisemblance de la loi
// imax = indice du maximum,  m = le maximum
```

```
  p=1/2;
  printf('N: indice du max -> maximum\n');
  for N=[10,100,1000] do
    loi=calculer_loi(N,p);
    QUESTION: [m,imax]= ON CHERCHE LE MAX DE LA LOI ET L'INDICE DU MAX
    imax=imax-1;// A cause du déclage de 1
    printf('%d: %d -> %f\n',N,imax,m);
  end
endfunction
```

2. Vérifier que la moyenne du "run maximum" varie linéairement en fonction de $\log(N)$.

```
function s=moyenne(loi)
  N=size(loi,'*')-1;
  QUESTION: s = CALCULER LA MOYENNE D'UNE V.A. DE LOI 'loi' ?
endfunction
```

```
function main_5_bis()
// La moyenne varie (approximativement) comme C log(N).
// Ca peut se prouver.
  p=1/2;
  i=0;
  for N=[10,50,100,500,1000,5000,10000] do
    i=i+1;
    loi=calculer_loi(N,p);
    x(i)=log(N);
    y(i)=moyenne(loi);
    printf('%d: %f ?=? %f\n',N,y(i),x(i));
  end;
  plot2d(x,y);
endfunction
```

2 Calcul du prix d'une option européenne dans le modèle de Cox-Ross

Simulation du modèle

On considère le chaîne de Cox-Ross :

$$X_0 = x_0, X_{n+1} = X_n \left(d\mathbf{1}_{\{U_{n+1} = P\}} + u\mathbf{1}_{\{U_{n+1} = F\}} \right).$$

avec $N = 10$, $x_0 = 100$, $p = 1/2$, $u = 1 + 1/N$, $d = 1 - 1/N$.

On cherche à calculer $\mathbf{E}(f(X_N))$ où $f(x) = \max(x - K, 0)$ avec $K = 100$.

1. Simuler cette chaîne de Markov.

```
function X=simul_cox_ross(N,x_0,p,u,d)
  U=grand(1,N,"bin",1,p);// tirages a pile ou face (p,1-p)
  X=x_0 * [1, cumprod(u^U .* d^(1-U))];
  // Plus long mais plus comprehensible ...
  // X(1)=x_0;
  // for i=1:prod(length(U)) do
  //  if U(i) then
  //    X(i+1)=X(i)*u
  //  else
  //    X(i+1)=X(i)*d;
  //  end
  //end;
endfunction;
```

```
function main_6()
  N=50;
  sigma=0.3;
  p=1/2;u=1-sigma/sqrt(N);d=1+sigma/sqrt(N);
  x_0=1;

  X=simul_cox_ross(N,x_0,p,u,d);
  plot2d2(0:N,X);
endfunction
```

Une version récursive

1. Ecrire une version récursive de l'algorithme de calcul de prix.

```
K=100;
function res=f(x)
  res=max(x-K,0);
endfunction
```

```
function res=prix_recursif(x,k,N)
  if k==N then res=f(x); return; end;
  QUESTION: res = RECOPIER L'EQUATION DU COURS !;
endfunction

function res=prix_slow(x,N)
  res=prix_recursif(x,0,N);
endfunction
```

2. Tester l'algorithme avec N petit ($N = 10$).

```
N=10;

// On choisit des paramètres pour converger
// vers le modèle de Black et Scholes.
sigma=0.3;
p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);

prix_slow(100,N);// Faire N=20 pour savoir ce que slow veut dire !
```

Une version itérative

1. Ecrire une version efficace (itérative) de l'algorithme de calcul de prix. Tracer la fonction $x \to u(0, x)$ pour $x \in [80, 120]$.

```
function res=inc(n);
// Permet de faire comme si les tableaux était indicés
// à partir de 0 (et non de 1)
 res=n+1;
endfunction;
```

```
function res=prix(x_0,N)
  // U=zeros(inc(N),inc(N));
  U=zeros(N+1,N+1);
  for k=[0:N] do
    // U(inc(N),inc(k)) = f(x_0 * u^k * d^(N-k));
    U(N+1,k+1) = f(x_0 * u^k * d^(N-k));
  end;

  for n=[N-1:-1:0] do // le temps decroit de N-1 a 0
    // ATTENTION AU DECALAGE DE 1
    QUESTION: U(n+1,1:n+1)=RECOPIER L'EQUATION DU COURS;
    // Une version "vectorisée" qui fait la même chose que
    //     for k=[0:n] do
    //       U(inc(n),inc(k)) = ...
    //                  p*U(inc(n+1),inc(k+1))+(1-p)*U(inc(n+1),inc(k));
    //     end;
  end;
  // res=U(inc(0),inc(0));
  res=U(1,1);
endfunction;
```

2. Comparer le résultat des deux versions de l'algorithme.

```
function main_7()
  N=10;
  sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  K=100;x_0=100;

  prix(x_0,N)

  // Les deux algos font ils le même chose ?
  // on verifie : prix_slow(x_0,N) \approx prix(x_0,N)
  printf('différence entre les 2 resultats: %e\n', ...
          abs(prix_slow(x_0,N) - prix(x_0,N)));
endfunction
```

3. Que constatez vous lorsque N augmente ($N = 10, 100, 200, 500$) et que l'on choisit u et d en fonction de N de la façon suivante :

$$u = 1 + \frac{\sigma}{\sqrt{N}} \text{ et } d = 1 - \frac{\sigma}{\sqrt{N}}.$$

```
function main_8()
// Avec cet algorithme on peut augmenter N
// mais il faut renormaliser convenablement u et d pour
// rester borné.
// Essayer avec N=10,100,200,...,1000
  sigma=0.6;
  couleur=1

  plot2d([50:150],max([50:150]-K,0));//xclick;

  for N=[3,5,10,20,50,100,200] do
    d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);

    n=0;
    for x=[50:150] do
      n=n+1;
      courbe(n)=prix(x,N);
    end

    couleur=couleur+1;
    plot2d([50:150],courbe,style=couleur);
    xclick;
  end
  // Ca converge, mais vers quoi ?
endfunction
```

Un cas numériquement délicat : les options sur moyenne

On cherche maintenant à évaluer $\mathbf{E}(f(S_N))$ où $S_n = X_1 + \cdots + X_n$.

1. Pourquoi le processus $(S_n, n \geq 0)$ n'est il pas une chaîne de Markov ? Vérifier que le couple $((X_n, S_n), n \geq 0)$ est une chaine de Markov de matrice de transition (0 sinon)

$$P((x,s),(xu,s+xu)) = p, \quad P((x,s),(xd,s+xd)) = 1-p.$$

issue de $(x_0, 0)$ à l'instant 0. En déduire que $\mathbf{E}(f(S_N)) = u(0, x_0, 0)$ où est la solution unique de

$$\begin{cases} u(n,x) = pu(n+1, xu, s+xu) + (1-p)u(n+1, xd, s+xd), & n < N \\ u(N,x,s) = f(s), \end{cases} \tag{1}$$

2. Ecrire un algorithme récursif (lent) qui résoud (1) ($N \leq 10$!) et permet de calculer $\mathbf{E}(f(S_N))$.

```
function res=f_moy(x,s); res=max((s/N)-K,0);endfunction

function res=prix_moyenne(x,s,k,N)
  if k==N then res=f_moy(x,s); return; end;
  QUESTION: res = <À COMPLÉTER>;
endfunction

function res=prix_slow_moyenne(x,N)
  res=prix_moyenne(x,x,0,N);
endfunction
```

3. Vérifier (par simulation ou à l'aide de l'algorithme précédent) que les points que peut atteindre la chaîne (X_n, S_n) sont tous différents et se convaincre qu'il sera (donc) difficile d'écrire un algorithme exact plus efficace que l'algorithme précédent pour calculer $\mathbf{E}(f(S_N))$.

```
function main_9()
  N=10;sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  x_0=100;K=100;

  // Ca marche mais ce n'est pas très efficace ...
  printf('Prix option sur moyenne: %f\n',prix_slow_moyenne(x_0,N));
endfunction
```

```
function liste=liste_moyenne_rec(x,s,k,N)
// On constitue la liste des points visités par la chaine
// Si un point est visité deux fois, il y figure 2 fois.
  if k==N then
    liste=[x;s];    //printf("%d: %f %f\n",x,s);
    return;
  end;
  QUESTION: liste = <À COMPLÉTER>;
```

```
endfunction

function liste=liste_moyenne(x,N)
  // On part de (x,s=x) a l'instant 0
  liste=liste_moyenne_rec(x,x,0,N)
endfunction
```

```
function main_10()
  N=10;
  x_0=100;
  sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);

  liste=liste_moyenne(x_0,N);

  // Tri des points selon les valeurs de la somme.
  // Les valeurs de x peuvent etre egales, mais pas celle de s.
  // Nous allons le verifier.
  [y,p]=sort(liste(2,:));
  liste1=liste(:,p);

  // On regarde si tous les points sont differents
  // en parcourrant le tableau ainsi classé
  epsilon=0.001;
  Taille=size(liste);
  match=[];
  for i=[1:Taille(2)-1] do
    if (norm(liste1(:,i)-liste1(:,i+1)) < epsilon) then
      printf("Warning: (%f,%f) ~ (%f,%f)\n",...
                 liste1(1,i),liste1(2,i),liste1(1,i+1),liste1(2,i+1));
      match= ...
        [match,[liste1(1,i),liste1(2,i),liste1(1,i+1),liste1(2,i+1)]];
    end;
  end;
  if size(match,'*') == 0 then
     printf("Aucun point n''est dupliqué.");
  end
endfunction
```

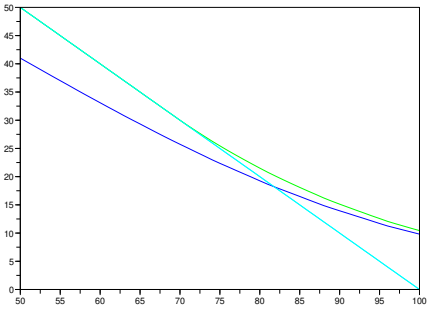
Dans ce cas l'"arbre n'est pas recombinant" et l'on ne peut éviter un algorithme de complexité 2^N. Il faut recourrir à d'autres approximations pour obtenir un algorithme réaliste (mais approché ...).

Temps d'arrêt, Arrêt optimal

Recrutement optimal, Options américaines

Bernard Lapeyre

http://cermics.enpc.fr/~bl



Plan

PROCESSUS DE MARKOV

- *Processus de Markov* et *propriété de Markov*

$$P\left(X_{n+1}=x_{n+1}|X_n=x_n, X_{n-1}=x_{n-1},\dots,X_0=x_0\right) = \mathbb{P}\left(X_{n+1}=x_{n+1}|X_n=x_n\right).$$

- la matrice de transition : $(P(x,y), x, y \in E)$

$$P(x,y) = \mathbb{P}\left(X_{n+1}=y|X_n=x\right).$$

- Loi de $(X_0,\dots,X_N)$ déterminée par P et la loi de X_0 (notée μ_0).

$$\mathbb{P}\left(X_0=x_0,\dots,X_n=x_n\right) = \mu_0(x_0)P(x_0,x_1)\times\cdots\times P(x_{n-1},x_n).$$

- Loi de X_N, μ_N donnée par $\mu_N = \mu_0 P^N$.

LOI DE X_n : UNE AUTRE FAÇON D'ÉCRIRE LES CHOSES

- $(X_n, n \geq 0)$ chaîne de Markov de matrice de transition P sur E.
- $\mathbb{P}(X_0 = x_0) = 1$ (i.e. $\mu_0(x_0)=1$, $\mu_0(x)=0$ si $x \neq x_0$.)
- On sait exprimer $\mathbb{E}(f(X_N))$

$$\mathbb{E}(f(X_N)) = \mu_N f = \mu_0 P^N f = \sum_{x\in E}\mu_0(x)(P^N f)(x) = (P^N f)(x_0).$$

Une formulation alternative plus algorithmique.

Theorem 1

Soit $(u(n,x), n=0,\dots,N, x\in E)$ la solution unique de

$$(1)\qquad \begin{cases} u(n,x) = \sum_{y\in E} P(x,y)u(n+1,y), & n<N \\ u(N,x) = f(x), \end{cases}$$

Alors $\mathbb{E}(f(X_N)) = u(0,x_0)$.

REMARQUES

- La première équation de (1) peut aussi s'écrire

$$u(n,x) = P[u(n+1,.)](x) = \sum_{y\in E} P(x,y)u(n+1,y)$$

- $u(n,x)$ peut s'interpréter comme

$$u(n,x) = \overbrace{P\times\cdots\times P}^{N-n \text{ fois}} f(x) = P^{N-n}f(x),$$

- Lorsque $\mathbb{P}(X_n = x) > 0$, on a

$$u(n,x) = \mathbb{E}\left(f(X_N)|X_n = x\right) = P^{N-n}f(x).$$

Preuve

$$\begin{aligned}&\mathbb{P}(X_0 = x_0, \cdots, X_n = x_n, X_{n+1} = x_{n+1}, \cdots, X_N = x_N)\\ &\quad = \mathbb{P}(X_0 = x_0)P(x_0, x_1) \times \cdots \times P(x_n, x_{n+1}) \times P(x_{n+1}, x_{n+2}) \times \cdots \times P(x_{N-1}, x_N)\end{aligned}$$

en sommant sur toutes les valeurs de $x_0, \cdots, x_{n-1}$ et $x_{n+1}, \cdots, x_{N-1}$ on obtient la loi de (X_n, X_N)

$$\mathbb{P}(X_n = x_n, X_N = x_N) = \mathbb{P}(X_n = x_n)P^{N-n}(x_n, x_N)$$

$$\mathbb{E}\left(f(X_N)\mathbf{1}_{\{X_n = x_n\}}\right) = \mathbb{P}(X_n = x_n)\sum_{x_N\in E} P^{N-n}(x_n, x_N)f(x_N) = \mathbb{P}(X_n = x_n)(P^{N-n}f)(x_n)$$

REMARQUES ET NOTATIONS

- (1) est *une équation de programmation dynamique* [1]
- E est fini, (1) est un *algorithme* qui termine.

UNE NOTATION COMMODE "À LA `Scilab`"

- $x_{0:n}$ plutôt que $(x_0, \ldots, x_n)$
- $X_{0:n}$ plutôt que $(X_0, \ldots, X_n)$
- $X_{0:n} = x_{0:n}$ plutôt que $(X_0 = x_0, \ldots, X_n = x_n)$

[1] Sur l'origine du terme programmation dynamique voir [Dreyfus(2002)]. "It was something not even a congressman could object to" selon Bellman.

Preuve formelle

On le sait déjà ... mais voici une autre méthode de preuve. On va montrer que

$$\mathbb{E}\left(u(n+1, X_{n+1})\right) = \mathbb{E}\left(u(n, X_n)\right).$$

Si cela est vrai :

$$u(0, x_0) = \mathbb{E}\left(u(0, X_0)\right) = \cdots = \mathbb{E}\left(u(N, X_N)\right) = \mathbb{E}\left(f(X_N)\right).$$

La loi de $X_{0:n+1} = (X_{0:n}, X_{n+1})$ est donnée par (c'est un façon d'exprimer la propriété de Markov)

$$\mathbb{P}\left(X_{0:n+1} = x_{0:n+1}\right) = \mathbb{P}\left(X_{0:n} = x_{0:n}, X_{n+1} = x_{n+1}\right) = \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}).$$

$$\begin{aligned}
\mathbb{E}\left(u(n+1, X_{n+1})\right) &= \sum_{x_{0:n+1} \in E^{n+2}} u(n+1, x_{n+1}) \mathbb{P}\left(X_{0:n+1} = x_{0:n+1}\right), \\
&= \sum_{x_{0:n} \in E^{n+1}, x_{n+1} \in E} u(n+1, x_{n+1}) \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}), \\
&= \sum_{x_{0:n} \in E^{n+1}} \mathbb{P}\left(X_{0:n} = x_{0:n}\right) \sum_{x_{n+1} \in E} P(x_n, x_{n+1}) u(n+1, x_{n+1}), \\
&= \sum_{x_{0:n} \in E^{n+1}} \mathbb{P}\left(X_{0:n} = x_{0:n}\right) u(n, x_n) = \mathbb{E}\left(u(n, X_n)\right).
\end{aligned}$$

La question du jour

- $(X_n, n \geq 0)$ une chaîne de Markov de matrice de transition P sur E. $\mathbb{P}(X_0 = x_0) = 1$.
- On cherche à calculer non pas $\mathbb{E}(f(X_N))$, mais $\sup_{\tau \leq N} \mathbb{E}(f(X_\tau))$.
- τ appartenant à une famille de temps aléatoires, *plus grande* que les temps déterministes, mais *plus petite* que tous les temps aléatoires.
- *plus grande* que les temps déterministes : parce que l'on souhaite pouvoir tenir compte des valeurs de X_n au fil du temps.
- *plus petite* que tous les temps aléatoires : parce que à l'instant n on ne connait pas la trajectoire future $X_{n+1}, \ldots, X_N$.
- $\tau = \text{Argmax}\{f(X_n), 0 \leq n \leq N\}$ est optimum, mais réclame de connaitre le futur!
- Voir Pour la Sciences [Hill(2009)] pour une introduction à ce genre de problème.

TEMPS D'ARRÊT

La notion adéquate est celle de *temps d'arrêt* : on souhaite prendre la décision d'arrêter avec l'information que l'on a au temps n.

Definition 2

τ est un *temps d'arrêt*, si, pour tout n, il existe $A_n \subset E^{n+1}$ tel que

$$\{\tau = n\} = \{(X_0, X_1, \dots, X_n) \in A_n\}$$

- "Je peux déterminer si $\tau = n$ en ne considérant que la portion de trajectoire avant n".
- Argmax$\{f(X_n), 0 \leq n \leq N\}$ ne peut pas être un temps d'arrêt (sauf cas particulier).
- Le temps d'atteinte d'un point z de E est un temps d'arrêt

$$\tau = \inf\{n \geq 0, X_n = z\}.$$

En effet $\{\tau = n\} = \{X_0 \neq z, X_1 \neq z, \dots, X_{n-1} \neq z, X_n = z\}$.

FORMULATION D'UN PROBLÈME D'ARRÊT OPTIMAL

- On se donne une chaîne de Markov $(X_n, n \geq 0)$ sur E, de matrice de transition P, issue de x_0 en 0.
- $f(n, x)$ donné, on cherche à calculer le sup suivant

$$\sup_{\tau \text{ t.a.} \leq N} \mathbb{E}(f(\tau, X_\tau))$$

- et aussi à identifier un τ qui réalise ce sup.
- ... on sait répondre complétement à ces deux questions.

Les données : P, x_0, f.

SOLUTION DU PROBLÈME D'ARRÊT OPTIMAL

Theorem 3

Si $(u(n,x), n = 0, \dots, N, x \in E)$ *est la solution unique de*

$$(2) \quad \begin{cases} u(n,x) = \max\left\{ \sum_{y\in E} P(x,y)u(n+1,y), f(n,x) \right\}, n < N, x \in E \\ u(N,x) = f(N,x), x \in E. \end{cases}$$

Alors

- $\sup_{\tau \le N} \mathbb{E}(f(\tau, X_\tau)) = u(0, x_0)$
- $\tau_0 = \inf\{n \ge 0, u(n, X_n) = f(n, X_n)\}$ *est un temps d'arrêt optimal.*

- Lorsque la matrice de transition dépend de n il faut remplacer P par P_n (la matrice de transition entre les instants n et $n+1$) dans l'équation.
- τ_0 est bien un temps d'arrêt $\le N$.

$$\{\tau_0 = n\} = \{u(0, X_0) \neq f(0, X_0), \dots, u(n-1, X_{n-1}) \neq f(n-1, X_{n-1}), u(n, X_n) = f(n, X_n)\}$$

- Preuve formelle du théorème : transparent 18.

COMMENTAIRES

- (2) est une équation de *programmation dynamique* proche de celle qui permet de calculer $\mathbb{E}(f(X_N))$.
- $u(n,x) = \sup_{n \le \tau \le N} \mathbb{E}(f(\tau, X_\tau) | X_n = x)$.
- Preuve informelle ("Principe d'optimalité") :

  En n, si $X_n = x$ *soit j'exerce en n et je gagne* $f(n,x)$ *soit j'attends* $n+1$ *où je peux gagner* $u(n+1, X_{n+1})$*, dont je dois calculer l'espérance en n sachant que* $X_n = x$ *qui est donnée par*

$$\sum_{y\in E} P(x,y)u(n+1,y).$$

- On l'utilise (2) pour écrire un algorithme (pas beaucoup plus compliqué (il suffit de rajouter le max) que celui qui permet de calculer $\mathbb{E}(f(X_N))$). Voir TD.

Exemple 1: Calcul du prix d'options américaines

- $(X_n, 0 \leq n \leq N)$ une chaîne de Markov de matrice de transition P décrivant l'évolution des prix des actifs (e.g. modèle de Cox-Ross).
- J'ai la possibilité si j'"exerce" en $n \leq N$ de gagner $f(n, X_n)$.
- Que vaut ce droit et à quel moment dois-je exercer ce droit pour maximiser mon gain ?
- Pour calculer le prix (i.e. la valeur de ce droit), il est naturel[2] de chercher à maximiser l'espérance du flux $\mathbb{E}(f(\tau, X_\tau))$ parmi tous les temps d'arrêt de X.

[2] pour une justification complète cf. cours de Mathématiques Financière (2A) ou [Lamberton and Lapeyre(1997)].

PROBLÈME CLASSIQUE : PUT AMÉRICAIN

- On cherche à calculer $\sup_{\tau \leq N} \mathbb{E}\left(f(\tau, X_\tau)\right)$, le prix de l'option et à determiner le moment d'exercice optimum.
- $(X_n, 0 \leq n \leq N)$ est le processus de Cox-Ross

$$X_0 = 1, X_{n+1} = X_n \left(u\mathbf{1}_{\{U_{n+1} = P\}} + d\mathbf{1}_{\{U_{n+1} = F\}} \right).$$

$(U_n, n \geq 1)$ une suite de tirage à pile ou face indépendant,

$$0 \leq p \leq 1, \quad \mathbb{P}(U_n = P) = p = 1 - \mathbb{P}(U_n = F).$$

- r le taux d'intérêt sur une période, K le strike, $d < 1 + r < u$.

$$f(n, x) = \frac{1}{(1+r)^n}(K - x)_+$$

SOLUTION

- On commence par calculer (cf TD) $(u(n, x), n = 0, \ldots, N, x \in E)$ la solution de l'équation (2) du théorème 3, ici

$$(3) \quad \begin{cases} u(n, x) = \max\left(p\, u(n+1, xu) + (1-p)\, u(n+1, xd), \dfrac{(K-x)_+}{(1+r)^n} \right), \\ u(N, x) = \dfrac{(K-x)_+}{(1+r)^N}. \end{cases}$$

- Souvent on calcule $v(n, x) = (1+r)^n u(n, x)$ plutot que $u(n, x)$.

$$\begin{cases} v(n, x) = \max\left(\dfrac{p\, v(n+1, xu) + (1-p)\, v(n+1, xd)}{1+r}, (K-x)_+ \right), \\ v(N, x) = (K-x)_+. \end{cases}$$

- $u(0, x) = v(0, x) = \sup_{\tau \leq N} \mathbb{E}\left(f(\tau, X_\tau)\right)$
- τ_0 le temps d'arrêt optimal

$$\begin{aligned} \tau_0 &= \inf\left\{n \geq 0, u(n, X_n) = (K - X_n)_+/(1+r)^n\right\} \\ &= \inf\left\{n \geq 0, v(n, X_n) = (K - X_n)_+\right\}. \end{aligned}$$

Preuve du théorème 3 : τ temps d'arrêt quelconque

- u solution (2), on va voir que, pour tout τ t.a., $\mathbb{E}\left(u\left(n \wedge \tau, X_{n\wedge\tau}\right)\right)$ **décroît en** n.
- En admettant ceci, on obtient

$$\begin{aligned} u(0,x_0) &= \mathbb{E}\left(u(0,X_0)\right) = \mathbb{E}\left(u\left(0\wedge\tau, X_{0\wedge\tau}\right)\right) \\ &\geq \mathbb{E}\left(u\left(N\wedge\tau, X_{N\wedge\tau}\right)\right) = \mathbb{E}\left(u\left(\tau, X_\tau\right)\right) \geq \mathbb{E}\left(f\left(\tau, X_\tau\right)\right). \end{aligned}$$

  Ce qui permet d'obtenir $\sup_{0\leq\tau\leq N, \text{t.a.}} \mathbb{E}\left(f\left(\tau, X_\tau\right)\right) \leq u(0,x_0)$.

- Pour montrer la décroissance annoncée, on remarque que

$$\begin{aligned} \Delta_{n+1} &= \mathbb{E}\left(u\left((n+1)\wedge\tau, X_{(n+1)\wedge\tau}\right)\right) - \mathbb{E}\left(u(n\wedge\tau, X_{n\wedge\tau})\right) \\ &= \mathbb{E}\left[\left(u(n+1, X_{n+1}) - u(n, X_n)\right)\mathbf{1}_{\{\tau \geq n+1\}}\right]. \end{aligned}$$

PREUVE DU THÉORÈME 3 : τ TEMPS D'ARRÊT QUELCONQUE

- τ est un temps d'arrêt, $\{\tau \geq n+1\} = \{\tau \leq n\}^c$, s'écrit sous la forme
$$\{\tau \geq n+1\} = \{X_{0:n} \in \bar{A}_n\}.$$
- La loi de $(X_{0:n}, X_{n+1})$ est donnée par (c'est la propriété de Markov)
$$\mathbb{P}\left(X_{0:n} = x_{0:n}, X_{n+1} = x_{n+1}\right) = \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}).$$

$$\begin{aligned}
\Delta_{n+1} &= \mathbb{E}\left[\left(u(n+1, X_{n+1}) - u(n, X_n)\right) \mathbf{1}_{\left\{X_{0:n} \in \bar{A}_n\right\}}\right], \\
&= \sum_{x_{0:n} \in \bar{A}_n, x_{n+1} \in E} \left(u(n+1, x_{n+1}) - u(n, x_n)\right) \mathbb{P}\left(X_{0:n} = x_{0:n}\right) P(x_n, x_{n+1}), \\
&= \sum_{x_{0:n} \in \bar{A}_n} \mathbb{P}\left(X_{0:n} = x_{0:n}\right) \left(\sum_{x_{n+1} \in E} u(n+1, x_{n+1}) P(x_n, x_{n+1}) - u(n, x_n) \right).
\end{aligned}$$

u sol. de (2), $\sum_{x_{n+1} \in E} u(n+1, x_{n+1}) P(x_n, x_{n+1}) \leq u(n, x_n)$, d'où $\Delta_{n+1} \leq 0$.

PREUVE DU THÉORÈME 3 : τ_0 TEMPS D'ARRÊT OPTIMAL

- $\tau_0 = \inf\{n \geq 0, u(n, X_n) = f(n, X_n)\}$.
- On va montrer que $\mathbb{E}\left(u\left(n \wedge \tau_0, X_{n \wedge \tau_0}\right)\right)$ **est constant en** n (i.e. $\Delta_{n+1} = 0$).
- En admettant ceci, il est facile de conclure
$$\begin{aligned}
u(0, x_0) &= \mathbb{E}\left(u(0, X_0)\right) = \mathbb{E}\left(u\left(0 \wedge \tau_0, X_{0 \wedge \tau_0}\right)\right), \\
&= \mathbb{E}\left(u\left(N \wedge \tau_0, X_{N \wedge \tau_0}\right)\right) = \mathbb{E}\left(u\left(\tau_0, X_{\tau_0}\right)\right),
\end{aligned}$$
Mais, par définition de τ_0, $u\left(\tau_0, X_{\tau_0}\right) = f\left(\tau_0, X_{\tau_0}\right)$, donc
$$u(0, x_0) = \mathbb{E}\left(f\left(\tau_0, X_{\tau_0}\right)\right).$$
- Ce qui finit la démonstration puisque τ_0 réalise alors le sup.
- Il nous reste à montrer que, pour ce temps d'arrêt $\Delta_{n+1} = 0$.

PREUVE DU THÉORÈME 3 : τ_0 TEMPS D'ARRÊT OPTIMAL

- τ_0 est un temps d'arrêt, $\{\tau_0 \geq n+1\} = \{X_{0:n} \in \bar{A}_n^0\}$ où

$$\bar{A}_n^0 = \{u(0,x_0) \neq f(0,x_0), \ldots, u(n,x_n) \neq f(n,x_n)\}\,.$$

- Sur l'événement $\{\tau_0 \geq n+1\}$, on a $u(n,X_n) \neq f(n,X_n)$ et comme u sol. de (2)

$$\sum_{x_{n+1}\in E} u(n+1,x_{n+1})P(X_n,x_{n+1}) = u(n,X_n).$$

- On termine alors comme tout à l'heure, mais en tenant compte de cette égalité :

$$\begin{aligned}\Delta_{n+1} &= \mathbb{E}\left[(u(n+1,X_{n+1}) - u(n,X_n))\,\mathbf{1}_{\left\{X_{0:n} \in \bar{A}^0{}_n\right\}}\right],\\ &= \sum_{x_{0:n}\in\bar{A}_n^0, x_{n+1}\in E} (u(n+1,x_{n+1}) - u(n,x_n))\,\mathbb{P}\left(X_{0:n} = x_{0:n}\right)P(x_n,x_{n+1}),\\ &= \mathbb{E}\left(\mathbf{1}_{\left\{X_{0:n} \in \bar{A}_n^0\right\}}\left(\sum_{x_{n+1}\in E} u(n+1,x_{n+1})P(X_n,x_{n+1}) - u(n,X_n)\right)\right) = 0.\end{aligned}$$

EXEMPLE 2: UN PROBLÈME DE RECRUTEMENT

- Je reçois, consécutivement, N candidats à un poste. Les circonstances m'imposent de décider tout de suite du recrutement (soit je recrute la personne que je viens de recevoir, soit je la refuse définitivement).
- La *seule* information que j'ai sur les candidats est leur classement.
- Je souhaite maximiser la probabilité de recruter le meilleur candidat.
- Quelle est la meilleure façon de s'y prendre ?

LE RÉSULTAT

- Il faut recevoir (environ) 37% des candidats, puis choisir le premier candidat qui suit qui est meilleur que tous les précédents (le dernier si cela n'arrive jamais).
- On obtient ainsi un temps d'arrêt optimal.
- La probabilité d'obtenir le meilleur candidat est (environ) de 37%.
- Les transparents qui suivent expliquent comment arriver à ces résultats à l'aide de la théorie précédente.

LE MODÈLE

- $\omega = (\omega_1, \ldots, \omega_N)$ une permutation de $(1, \ldots, N)$.
- ω_k le classement du $\#k$-ième individu dans la permutation.

$$\begin{array}{lccccccc} \textit{Indice} & (\#1 & \#2 & \ldots & \#k & \ldots & \#N) \\ \textit{Rang} & (\omega_1 & \omega_2 & \ldots & \omega_k & \ldots & \omega_N) \end{array}$$

- Ω_N l'ensemble des permutations de $(1, \ldots, N)$ muni de la probabilité uniforme.
- B_n l'évenement "le n-ième candidat est le meilleur".
- On cherche un temps d'arrêt τ qui maximise $\mathbb{P}(B_\tau)$.

OÙ EST LA CHAÎNE DE MARKOV ?

- Un temps d'arrêt mais pour quel processus de Markov ?
- $R_k(\omega)$ le rang du $\#k$-ième individu parmi les k premiers individus.
- $B_n = \{R_n = 1, R_{n+1} > 1, \ldots, R_N > 1\}$.
- Quelle est la loi de $(R_1, \ldots, R_N)$?

UN EXEMPLE DE CALCUL DE R

- $\omega = (2\,3\,1\,4)$ donne $R = (1\,2\,1\,4)$.
- $R = (1\,2\,1\,4)$ donne $\#3 \leq \#1 \leq \#2 \leq \#4$

$$
\begin{aligned}
(1) &\to (\#1) && \#1 \text{ est le premier puisqu'il est tout seul!}\\
(1\,2) &\to (\#1 \leq \#2) && \#2 \text{ est le deuxième parmi les 2 premiers}\\
(1\,2\,1) &\to (\#3 \leq \#1 \leq \#2) && \#3 \text{ est le premier parmi les 3 premiers}\\
(1\,2\,1\,4) &\to (\#3 \leq \#1 \leq \#2 \leq \#4) && \#4 \text{ est le quatrième parmi les 4 premiers}
\end{aligned}
$$

On obtient la permutation de départ $\omega = (2\,3\,1\,4)$

Indice	$(\#1$	$\#2$	$\#3$	$\#4)$
Rang	$(2$	3	1	$4)$

EN Python

Si vous avez un doute voici, deux fonctions `Python` qui font le job.

```
import numpy as np

def Omega2R(omega):
# Calcule les rang d'insertion pour un omega donne
  R=np.zeros(np.size(omega), dtype=int)
  # dtype=int, pour specifier que l'on veut un tableau d'entier
  for n in range(np.size(omega)):
    # classe le vecteur omega[1,...,n] en croissant
    y=np.sort(omega[0:n+1]);
    # R[n] = le classement de omega[n] parmi les n premiers
    R[n]=np.where(y==omega[n])[0][0]+1
  return R

def R2Omega(R):
    # Calcule omega connaissant les rangs d'insertion
    iomega=np.zeros(0, dtype=int);
    for n in range(np.size(R)):
        # J'insére n à l'indice R[n]
        iomega=np.concatenate([iomega[0:int(R[n]-1)],[n+1],iomega[int(R[n]-1):n+1]])
    # On inverse la permutation
    omega=np.zeros(np.size(R), dtype=int)
    for n in range(np.size(omega)):
        omega[iomega[n]-1]=n+1
    return omega

Taille=6
omega=np.random.permutation(Taille)+1 # permutation aléatoire
print(omega)
print(R2Omega(Omega2R(omega))) # devrait être égal à omega
```

Calcul de lois

- À un R correspond un et un seul ω, **donc**, pour $\alpha_k \in \{1, \dots, k\}$

$$P(R_1 = \alpha_1, \dots, R_N = \alpha_N) = \mathbb{P}(\{\omega\}) = \frac{1}{N!}.$$

- (Par contraction) les R_k suivent des lois uniformes sur $\{1, \dots, k\}$. Elles sont indépendantes.
- $S_k = \mathbf{1}_{\{R_k = 1\}}$ sont aussi des variables aléatoires indépendantes. Elles suivent des lois de Bernouilli de paramêtre $p_k = \mathbb{P}(S_k = 1) = 1/k$.
- La suite de variable $(S_1, \dots, S_N)$ suffira pour traiter le problème.

Le critère

- B_n l'évenement "le n-ième candidat est le meilleur".

$$B_n = \{R_n = 1, R_{n+1} > 1, \dots, R_N > 1\} = \{S_n = 1, S_{n+1} = 0, \dots, S_N = 0\}.$$

- τ un temps d'arrêt par rapport au processus $R = (R_1, \dots, R_N)$. À l'instant n, $(R_1, \dots, R_n)$ "contient toute l'information disponible".
- Pour un temps d'arrêt[3] τ, on va voir que

$$\mathbb{P}(B_\tau) = \mathbb{E}\left(\frac{\tau}{N} S_\tau\right).$$

ce qui permettra de mettre le problème "sous forme markovienne".

[3] Ce n'est plus vrai sinon ... Exercice: trouver un contre-exemple.

PREUVE

$$\{\tau = n\} = \{(R_1, \dots, R_{n-1}, R_n) \in A_n\} = \{R_{1:n} \in A_n\}.$$

Notation : $R_{1:n} = (R_1, \dots, R_{n-1}, R_n)$.

$$\begin{aligned}\mathbb{P}(\tau = n, B_\tau) &= \mathbb{P}(\tau = n, B_n).\\ &= \mathbb{P}(\tau = n, R_n = 1, R_{n+1} > 1, \dots, R_N > 1)\\ &= \mathbb{P}(R_{1:n} \in A_n, R_n = 1, R_{n+1} > 1, \dots, R_N > 1)\\ &= \mathbb{P}(R_{1:n} \in A_n, R_n = 1)\mathbb{P}(R_{n+1} > 1, \dots, R_N > 1)\end{aligned}$$

car indépendance des vecteurs $R_{1:n}$ et $R_{n+1:N}$, puis par indépendance des R_n entre eux

$$\mathbb{P}(R_{n+1} > 1, \dots, R_N > 1) = \frac{n}{n+1}\frac{n+1}{n+2} \times \cdots \times \frac{N-1}{N} = \frac{n}{N}$$

PREUVE

$$\begin{aligned}\mathbb{P}(\tau = n, B_\tau) &= \frac{n}{N}\mathbb{P}(R_{1:n} \in A_n, R_n = 1) = \frac{n}{N}\mathbb{P}(\tau = n, R_n = 1)\\ &= \frac{n}{N}\mathbb{E}\left(\mathbf{1}_{\{\tau = n, R_n = 1\}}\right) = \mathbb{E}\left(\mathbf{1}_{\{\tau = n\}}\frac{n}{N}S_n\right)\\ &= \mathbb{E}\left(\mathbf{1}_{\{\tau = n\}}\frac{\tau}{N}S_\tau\right).\end{aligned}$$

En sommant pour n variant de 1 à N

$$\mathbb{P}(B_\tau) = \mathbb{E}\left(\frac{\tau}{N}S_\tau\right).$$

RÉSUMONS NOUS !

- $(S_1, \dots, S_N)$ est une suite de variables aléatoires indépendantes de Bernouilli $1/k$, **donc** une chaîne de Markov sur l'espace $E = \{0, 1\}$, non homogène, de matrice de transition, dépendant du temps, P_n

$$P_n(0,0) = P_n(1,0) = 1 - \frac{1}{n+1} = \frac{n}{n+1}$$
$$P_n(0,1) = P_n(1,1) = \frac{1}{n+1}$$

- On chercher à maximiser $\mathbb{P}(B_\tau) = \mathbb{E}\left(\frac{\tau}{N} S_\tau\right) = \mathbb{E}\left(f(\tau, S_\tau)\right)$ parmi tous les temps d'arrêt.

RÉSOLUTION

- On peut résoudre le problème grâce à (2)
- On calcule (cf TD) $(u(n, 0 \text{ ou } 1), 0 \leq n \leq N)$

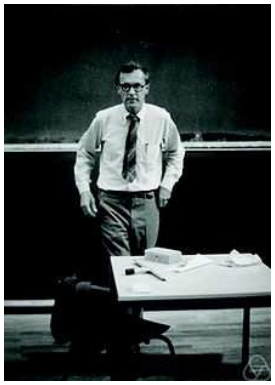
$$\begin{cases} u(n,x) = \max\left\{\frac{n}{n+1} u(n+1,0) + \frac{1}{n+1} u(n+1,1), \frac{n}{N} x\right\}, n < N, \\ u(N,x) = x, \end{cases}$$

- Une fois ceci fait, un temps d'arrêt optimal est obtenu par

$$\tau = \inf\left\{n \geq 0, u(n+1, S_n) = \frac{n}{N} S_n\right\}.$$

- Dans ce cas, on peut mener des calculs explicites (voir [Delmas and Jourdain(2006)] et TD) pour obtenir les résultats annoncés.

SNELL, JAMES LAURIE



- La théorie de l'arrêt optimal porte le nom d'enveloppe de Snell (voir [Snell(1952)]), "so named by the Russian mathematician, Kolmogorov".
- Snell (1925-2011), mathématicien américain, élève de Doob.
- Il utilise la théorie des martingales pour traiter le problème d'arrêt optimal.

BIBLIOGRAPHIE


Jean-François Delmas and Benjamin Jourdain.
Modèles aléatoires.
Mathématiques & Applications. Springer-Verlag, Berlin, 2006.


Stuart Dreyfus.
Richard bellman on the birth of dynamic programming.
Operations Research, 50(1):48–51, 2002.


Theodore Hill.
Savoir quand s'arrêter.
Pour La Science, 381, Juillet 2009.


Damien Lamberton and Bernard Lapeyre.
Introduction au calcul stochastique appliqué à la finance.
Ellipses, Édition Marketing, Paris, second edition, 1997.


J. L. Snell.
Applications of martingale system theorems.
Trans. Amer. Math. Soc., 73:293–312, 1952.

Cours de Décision dans l'incertain
Exercices : vendredi 22 mai 2020.

Exercice 1 On considère un dé à 6 faces équiprobables. Un joueur a le choix de lancer ce dé 1, 2 ou 3 fois, il obtient alors pour gain la valeur du dernier lancé effectué.

En supposant les 3 tirages indépendants, l'objectif de cet exercice est déterminer la stratégie optimale en moyenne de ce joueur.

1. Calculer l'espérance du gain si l'on effectue un seul tirage.
2. Si le joueur doit choisir un nombre de tirages fixe (déterministe), avant le début du jeu, a t'il intérêt à choisir entre jouer 1, 2 ou 3 fois ?
3. Quel est le gain optimal moyen du joueur, s'il connait les résultats des 3 tirages (le gain est alors le maximum des 3 tirages indépendants) ?

On suppose maintenant que le joueur peut tenir compte des tirages déjà effectués.

4. Montrer que la stratégie optimale (en moyenne), si l'on se restreint à 2 tirages maximum, est la suivante : si le premier tirage vaut 4, 5 ou 6 s'arrêter, sinon rejouer. Calculer la moyenne du gain de cette stratégie optimale.
5. En utilisant la question précédente identifier la stratégie optimale lorsque l'on joue 3 fois. Calculer la moyenne du gain de cette stratégie optimale.

Exercice 2 On considère une chaîne de Markov $(X_n, n \geq 0)$, sur un espace fini E, de matrice de transition P, issue d'un point $x_0 \in E$ (i.e. $\mathbb{P}(X_0 = x_0) = 1$).

On cherche $(u(n,x), n = 0, \ldots, N, x \in E)$ une solution à l'équation

$$\begin{cases} u(n,x) = \max\left\{ \sum_{y\in E} P(x,y)u(n+1,y), f(n,x)\right\}, n < N, x \in E \\ u(N,x) = f(N,x), x \in E. \end{cases} \tag{1}$$

Montrer qu'il existe une solution à cette équation (**??**) et que cette solution est unique.

Exercice 3 Avec les notation de l'exercice précèdent, on note u la solution unique de (**??**).

1. Montrer que pour tout temps d'arrêt τ, et pour tout n, il existe, un ensemble $\bar{A}_n \subset E^n$ tel que
$$\{\tau \geq n+1\} = \{\tau \leq n\}^c = \{X_{0:n} \in \bar{A}_n\}.$$
où $X_{0:n} = (X_0, \ldots, X_n)$.
2. Montrer que, quel que soit le temps d'arrêt τ plus petit que N, $\mathbb{E}\left(u\left(n \wedge \tau, X_{n\wedge\tau}\right)\right)$ est décroissant en n pour $0 \leq n \leq N$ (s'aider des transparents du cours au besoin).
3. En déduire que pour tout temps d'arrêt τ plus petit que N, $u(0,x_0) \geq \mathbb{E}\left(f\left(\tau, X_\tau\right)\right)$.
4. On note $\tau_0 = \inf\{n \geq 0, u(n, X_n) = f(n, X_n)\}$. Montrer que τ_0 est un temps d'arrêt plus petit que N.
5. Montrer que $\mathbb{E}\left(u\left(n \wedge \tau_0, X_{n\wedge\tau_0}\right)\right)$ ne dépend pas de n, pour $0 \leq n \leq N$ (s'aider des transparents du cours au besoin).
6. En déduire que $u(0,x_0) = \mathbb{E}\left(f\left(\tau_0, X_{\tau_0}\right)\right) = \sup_{0\leq\tau\leq N, \text{t.a.}} \mathbb{E}\left(f\left(\tau, X_\tau\right)\right)$.

Exercice 4 On considère une chaîne de Markov $(X_n, n \geq 0)$ sur l'espace d'état $\{1, 2\}$ de matrice de transition P $(0 < p < 1)$

$$P = \begin{pmatrix} p & 1-p \\ p & 1-p \end{pmatrix},$$

issue de 1 à l'instant 0 (i.e. $\mathbb{P}(X_0 = 1) = 1$).

1. Montrer que, si $n \geq 1$, la loi de X_n est donnée par $\mathbb{P}(X_n = 1) = p$ et $\mathbb{P}(X_n = 2) = 1-p$.
2. En déduire que $(X_n, n \geq 1)$ est une suite de variables aléatoires indépendantes de même loi.
3. On s'intéresse au problème d'arrêt optimal pour cette chaîne de Markov avec comme gain à l'instant n, $f(n, x) = \rho^n \mathbf{1}_{\{x=1\}}$. On suppose que $\rho > 0$ et $p\rho > 1$. Montrer que la solution u de l'équation (**??**) est donnée, pour $n < N$, par $u(n, 1 \text{ ou } 2) = p\rho^N$ et pour $n = N$ par $u(N, x) = \rho^N \mathbf{1}_{\{x=1\}}$.
4. En déduire que le temps d'arrêt τ_{opt} optimal (unique dans ce cas) est donné par $\tau_{opt} = N$. Comment interpréter ce résultat ?
5. On pose $\bar{\tau}_{opt} = \sup\{n \geq 0, X_n = 1\}$. Vérifier que $\bar{\tau}_{opt}$ est l'unique temps aléatoire tel que
$$f(\bar{\tau}_{opt}, X_{\bar{\tau}_{opt}}) = \arg\max\{0 \leq n \leq N, f(n, X_n)\}.$$
6. Vérifier que $\mathbb{E}\left(f(\bar{\tau}_{opt}, X_{\bar{\tau}_{opt}})\right) > \mathbb{E}\left(f(\tau_{opt}, X_{\tau_{opt}})\right)$. $\bar{\tau}_{opt}$ peut-il être un temps d'arrêt ?

Chaîne de Markov : deux problèmes d'arrêt optimal

Bernard Lapeyre
CERMICS, École des Ponts ParisTech

3 mars 2020

Table des matières

Les programmes suivant on été prévu pour être exécutés à l'aide `Scicoslab`. Ce programme peut être téléchargé librement sur `www.scicoslab.org`. Il est disponible sur MacOSX, Windows ou Linux (Debian, Ubuntu ou Fedora). Noter toutefois qu'il faut installer le server `X`, `XQuartz` sur MacOSX. Ces programmes peuvent aussi fonctionner, moyennant de légère modification, avec la version officielle de `Scilab`.
Le fichier source en `Scilab` correspondant à ce document est accessible. Il est partiellement mais pas totalement corrigé. Le correction complète sera accessible à la fin du TD.

1 Le calcul du prix d'une option "américaine"

Rappel : le cas européen (calcul d'espérance)

On considère le modèle de Cox-Ross :

$$X_0 = x_0, X_{n+1} = X_n \left(d\mathbf{1}_{\{U_{n+1} = P\}} + u\mathbf{1}_{\{U_{n+1} = F\}} \right).$$

On choisit les valeurs numériques de la façon suivante

$$x_0 = 100, r_0 = 0,1, \sigma = 0,3.$$

et l'on définit p, r, u et d en fonction de N de la façon suivante :

$$p = 1/2,\ r = r_0/N,\ u = 1 + \frac{\sigma}{\sqrt{N}} \text{ et } d = 1 - \frac{\sigma}{\sqrt{N}}.$$

On cherche à évaluer $\mathbf{E}(f(N, X_N))$ où

$$f(N,x) = \frac{1}{(1+r)^N} \max(K - x, 0),$$

avec $K = x_0 = 100$ et $r = 0,05$.

1. Calculer ces prix d'options européennes (call et put) se ramène à des calculs d'espérance d'une fonction d'une chaîne de Markov. On implémente ici la méthode de calcul d'espérance par "programmation dynamique".

```
// Payoff du put
function res=gain_put(x,K); res=max(K-x,0);endfunction

// Payoff du call
function res=gain_call(x,K); res=max(x-K,0);endfunction

// Une fonction pour gérer le décalage de 1 dans les vecteurs
function res=inc(n); res=n+1; endfunction;

function res=prix_eu(x_0,N,gain)
// Calcul du prix europeen a l'instant 0
  // ATTENTION AU DECALAGE DE 1 EN TEMPS ET EN ESPACE
  U=zeros(inc(N),inc(N));
  for k=[0:N] do
    U(inc(N),inc(k)) = gain(x_0 * u^k * d^(N-k),K)/(1+r)^N;
  end;
  for n=[N-1:-1:0] do // le temps decroit de N-1 a 0
    U(inc(n),inc(0):inc(n)) =  ...
        p*U(inc(n+1),inc(1):inc(n+1))+(1-p)*U(inc(n+1),inc(0):inc(n));
  end;
  res=U(inc(0),inc(0));
endfunction;

function res=prix_eu_n(n,x_0,N,gain)
// Calcul du prix a l'instant n
  res=prix_eu(x_0,N-n,gain);
endfunction
```

2. On peut vérifier que lorsque $p = 1/2$ (ou $r = 0$) et $x = K$ le prix des puts et des calls coïncident. Pour le choix classique $(1+r-d)/(u-d)$ (et $r \neq 0$), les prix sont différents, ce que l'on vérifie aussi.

```
function main_1()
  r_0=0.05;sigma=0.3;
  x_0=100;K=100;

  N=50;
  d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  r=r_0/N;

  // Lorsque p=1/2 et K=x_0 le prix du call = le prix du put (exercice!)
  p=1/2;K=x_0;
  p_put  = prix_eu_n(0,x_0,N,gain_put);
  p_call = prix_eu_n(0,x_0,N,gain_call);
  if abs(p_put - p_call) >= 1000 * %eps then
     printf("WARNING: ces deux prix devrait coincider: %f <> %f\n",...
            p_put,p_call);
  else
     printf("Parfait!\n");
  end
```

```
  p= (1+r-d)/(u-d);
  printf("Prix du call : %f\n",prix_eu_n(0,x_0,N,gain_call));
  printf("Prix du put : %f\n",prix_eu_n(0,x_0,N,gain_put));
endfunction
```

3. Voici un programme qui permet de calculer des histogrammes. Si `varargin` est vrai un dessin est tracé, sinon seul l'histogramme est retourné dans `H`.

```
function H=histo_discret(samples, maxsize)
// histogramme de tirages selon une loi discrète à valeurs entières
// supposé prendre des valeurs entre 0 et max.
// Si varargin=\%t on trace l'histogramme.
  H=0
  for k=0:maxsize
    // Calcul du nbre de tirages valant k / Taille
    H(k+1)=length(find(samples==k)) ./ size(samples,'*');
  end;

  xbasc;plot2d3(0:maxsize,H);
  f=gcf();
  Dessin=f.children(1).children(1).children(1);
  Dessin.thickness=10;Dessin.foreground=5;
endfunction
```

Le cas américain (arrêt optimal)

On s'intéresse au cas d'un option américaine qui promet (en valeur actualisée en 0), si on l'exerce à l'intant n pour une valeur de X_n valant x

$$\frac{1}{(1+r)^n}\max(K-x,0).$$

On cherche à calculer son prix donné par

$$\sup_{\tau \text{t.a.} \leq N} \mathbf{E}(f(\tau, X_\tau)).$$

4. On pose $v(n,x) = (1+r)^n u(n,x)$, vérifier en utilisant l'équation qui définit u dans le cours, que v est solution de

$$\begin{cases} v(n,x) = \max\left[\dfrac{pv(n+1,xu)+(1-p)v(n+1,xd)}{1+r}, (K-x)_+\right], n < N, x \in E \\ v(N,x) = (K-x)_+, x \in E. \end{cases} \tag{1}$$

5. Ecrire un algorithme récursif (et inefficace ...) pour le calcul de v en recopiant l'équation (1)

```
function res=prix_recursif_am(x,k,N,gain)
  if k==N then res=gain(x); return; end;
  QUESTION: res = <À COMPLÉTER>
  res = max( res / (1+r), gain(x) );
endfunction

function res=prix_slow_am(x,N,gain)
  res=prix_recursif_am(x,0,N,gain);
endfunction
```

6. Ecrire un algorithme efficace de calcul de $v(n,x)$ (le prix de l'option américaine en n).

```
function res=prix_am(x_0,N,gain)
// Calcul du prix americain a l'instant 0
  U=zeros(N+1,N+1);
  x=x_0 * u^[0:N] .* d^[N:-1:0]; // les N+1 points de calcul en N
  U(N+1,1:N+1)=gain(x);// Valeur de U en N

  // ATTENTION AU DECALAGE DE 1 en espace et en temps
  for n=[N-1:-1:0] do // le temps decroit de N-1 a 0
    U(n+1,1:n+1)=p*U(n+2,2:n+2) + (1-p)* U(n+2,1:n+1);
    U(n+1,1:n+1)= U(n+1,1:n+1) / (1+r);// actualisation
    x = x_0 * u^[0:n] .* d^[n:-1:0]; // les points de calcul en n
    QUESTION: U(n+1,1:n+1)= POUR UNE OPTION AMERICAINE ?
  end;
  res=U(1,1);
endfunction;

function res=prix_am_n(n,x_0,N,gain)
  res=prix_am(x_0,N-n,gain);
endfunction
```

7. Pour $N = 10$, comparer les 2 méthodes pour vérifier que tout fonctionne. Effectuer des calculs de prix avec des N plus grands (uniquement pour la deuxième méthode, bien sûr).

```
function main_2()
  sigma=0.3; r_0=0.1;
  K=100;x_0=100;

  N=10;
  r=r_0/N;
  d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  p= (1+r-d)/(u-d);//p=1/2;


  prix_am(x_0,N,gain_put)
  prix_slow_am(x_0,N,gain_put)

  // Les deux algos font ils le même chose ?
  // on verifie : prix_slow(x_0,N) \approx prix(x_0,N)
  p1=prix_slow_am(x_0,N,gain_put);
  p2=prix_am(x_0,N,gain_put);
```

```
  if (abs(p1 - p2) >= 10 * %eps) then
     printf("WARNING: ces deux prix devrait coincider: %f <> %f\n",...
            p1,p2);
  else
     printf("Parfait!\n");
  end

  N=1000;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);

  prix_am(x_0,N,gain_put)
endfunction;
```

8. Tracer les courbes de prix américaines et européennes $x \to v(0,x)$ pour $x \in [80, 120]$ et les supperposer au "payoff".

```
function main_3()
  // tracé de courbes
  N=50;
  sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  K=100;x_0=100;
  r_0=0.1;
  r=r_0/N;

  vmin=50;
  vmax=150;

  n=0;
  for x=[vmin:vmax] do
    n=n+1;
    QUESTION: courbe_am(n) = <A COMPLÉTER>;
    QUESTION: courbe_eu(n) = <A COMPLÉTER>;
  end
  // On compare les courbes "Américaines" et "Européennes"
  plot2d([vmin:vmax],courbe_eu,style=2);
  plot2d([vmin:vmax],courbe_am,style=3);
  plot2d([vmin:vmax],gain_put([vmin:vmax]),style=4);
endfunction;
```

9. Regardez comment le prix évolue au fils du temps : tracez les courbes $x \to v(n,x)$, pour $n = 10, 20, 30, 40, 50$.

```
function main_4()
  sigma=0.3;
  K=100;x_0=100;

  N=50;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  r_0=0.1;r=r_0/N;

  vmin=50;vmax=150;
```

```
  // évolution de la courbe en fonction de n
  for n=[0,10,20,30,40,45,47,49,50] do
    i=0;
    for x=[vmin:vmax] do
      i=i+1;
      QUESTION: courbe_am(i) = <A COMPLÉTER>;
    end
    plot2d([vmin:vmax],courbe_am);
  end
endfunction
```

10. Que constatez vous lorsque N augmente ($N = 10, 100, 200, 500$) et que l'on choisit r, u et d en fonction de N comme définis plus haut.

Commentaire : lorsque N vers $+\infty$ dans ces conditions, on converge vers le prix dans un modèle continu (et célèbre : le modèle de Black et Scholes). Dans le cas européen, le résultat se prouve grâce au théorème de la limite centrale. Dans le cas américain, c'est un peu plus compliqué !

```
function main_5()
  // Avec cet algorithme on peut augmenter N
  // mais il faut renormaliser convenablement u et d.
  // Essayer avec N=10,100,200,...,1000

  sigma=0.3;
  K=100;x_0=100;

  r_0=0.1;
  p=1/2;
  N=50;

  // La renormalisation convenable
  // pour converger vers un modèle de Black et Scholes
  r=r_0/N;
  d=1-sigma/sqrt(N);
  u=1+sigma/sqrt(N);

  vmin=50;vmax=150;
  for N=[5,10,20,30,50] do
    r=r_0/N;
    d=1-sigma/sqrt(N);
    u=1+sigma/sqrt(N);
    n=0;
    for x=[vmin:vmax] do
      n=n+1;
      courbe(n)=prix_am(x,N,gain_put);
    end
    plot2d([vmin:vmax],courbe);
    plot2d([vmin:vmax],max(K-[vmin:vmax],0));// Le payoff / l'obstacle
  end
endfunction
```

11. Ecrire une fonction qui simule le vecteur $(Bin(1),\dots,Bin(N))$ du nombre de pile cumulé dans des N tirages à pile ou façe $(p,1-p)$.
Par la suite le valeur $X(n)$ du modèle de Cox-Ross à l'instant n s'écrira :

$$X(n) = x_0 u^{Bin(n)} d^{n-Bin(n)}.$$

```
function res=simul_bin(N,p)
// sommes partielles nombre de N tirages a pile ou face (p,1-p)
  bin=grand(1,N,"bin",1,p);// tirages a pile ou face (p,1-p)
  QUESTION: res=<A COMPLÉTER>;// sommes partielles
endfunction;
```

12. Calculer la prix américain en conservant dans un vecteur `V(t,k)` les valeurs en l'instant $t-1$ et au point $X(n) = x_0 u^{k-1} d^{n-(k-1)}$. Le décalage de 1 est du, comme d'habitude, au fait que les tableaux sont indicés à partir de 1 en ScicosLab.

```
function V=prix_am_vect(x_0,N,gain)
// On calcule les valeurs de "v(n,x)" (voir question précédente)
// mais, ici, on les conservent dans un vecteur "V"
// ce qui évitera d'avoir à les re-calculer un grand
// nombre de fois

  // ATTENTION AU DECALAGE DE 1 en ESPACE ET EN TEMPS
  V=zeros(N+1,N+1);
  x=x_0 * u^[0:N] .* d^[N:-1:0]; // les points de calcul en N
  V(N+1,1:N+1)=gain(x);

  for n=[N-1:-1:0] do // le temps décroit de N-1 a 0
    V(n+1,1:n+1)=p*V(n+2,2:n+2) + (1-p)* V(n+2,1:n+1);
    V(n+1,1:n+1)= V(n+1,1:n+1) / (1+r);// actualisation
    x=x_0 * u^[0:n] .* d^[n:-1:0]; // les points de calcul en  n
    V(n+1,1:n+1)=max(V(n+1,1:n+1),gain(x));
  end;
endfunction;
```

13. A partir d'une trajectoire du modèle de Cox-Ross donnée par le vecteur Bin et des valeurs de V précédemment calculées, évaluer le temps d'arrêt optimal associé à cette trajectoire.

```
function res=temps_optimal_option(Bin,V,gain)
  // Calcule le temps d'arrêt optimal associé à la trajectoire
  // (X(1),...,X(N)) où  X(n) =  x_0*u^Bin(n)*d^(n-Bin(n))
  // Bin est la somme cumulée de tirages de Bernouilli indépendants
  if gain(x_0) == V(1,1)  then  res=0;return;end;
  for n=1:size(Bin,'*')-1 do
    x=x_0*u^Bin(n)*d^(n-Bin(n));// Valeur de X(n)
    QUESTION: if (ÉCRIRE LA CONDITION D'ARRET & (gain(x) > 0) then
      res=n;return;
```

```
    end;
  end;
  res=N;
endfunction
```

14. Simuler un grand nombre de trajectoires du modèle de Cox-Ross et les valeurs des temps d'arrêt associés à ces trajectoires. Tracer un histogramme de la loi du temps d'arrêt optimal. Faire varier les valeurs de x_0 ($x_0 = 60, 70, 80, 100, 120$)) et voir l'influence de ce choix sur le loi du temps d'arrêt optimal.

```
function tau=un_test(x_0,N)
  sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  K=100;
  r_0=0.1;
  r=r_0/N;

  V=prix_am_vect(x_0,N,gain_put);
  Nbre=1000;
  for j=1:Nbre do
    Bin=simul_bin(N,p);
    tau(j)=temps_optimal_option(Bin,V,gain_put);
  end
  histo_discret(tau,max(tau));
endfunction;

function main_6()
  // un test simple "at the money" (K = x_0)
  N=100;
  x_0=100;tau=un_test(x_0,N);
endfunction;

function main_7()
  N=50;
  x_0=60;tau=un_test(x_0,N);// on exerce toujours en 0
  x_0=70;tau=un_test(x_0,N);// on n'exerce plus (jamais) en 0
  x_0=80;tau=un_test(x_0,N);// On exerce de plus en plus tard ...
  x_0=100;tau=un_test(x_0,N);
  x_0=120;tau=un_test(x_0,N);// Le plus souvent en N
endfunction;
```

15. Traiter le cas du call. On peut montrer qu'il ne s'exerce qu'à l'échéance. Le vérifier par simulation.

```
// Teste le cas du call (x-K)_+
// qui ne s'exerce jamais avant l'écheance N.
// Le fait que le prix eu du call soit toujours plus grand
// que l'obstacle permet de le prouver rigoureusement.

function main_8()
  function res=gain_call(x);
```

```
  // Payoff du call
    res=max(x-K,0);
  endfunction

  sigma=0.3; r_0=0.1;
  K=100;x_0=100;

  N=50;
  r=r_0/N;
  u=(1+r)*exp(sigma/sqrt(N));d=(1+r)*exp(-sigma/sqrt(N));
  p=1/2;  //p= (1+r-d)/(u-d); // en principe pour Cox-Ross

  V_call=prix_am_vect(x_0,N,gain_call);
  Nbre=1000;
  for j=1:Nbre do
    Bin=simul_bin(N,p);
    tau(j)=temps_optimal_option(Bin,V_call,gain_call);
  end
  histo_discret(tau,max(tau));// s'exerce toujours en N
endfunction;
```

16. Évaluer la probabilité d'exercice anticipé (i.e. $\mathbf{P}(\tau < N)$, si τ est le temps d'arrêt optimal. Vérifier par simulation que cette probabilité est strictement positive (pour le put) si $r > 0$.

```
function proba=compute_proba(x_0,N)
  // Proba que tau<N
  sigma=0.3;
  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);
  K=100;
  r_0=0.1;
  r=r_0/N;

  V=prix_am_vect(x_0,N,gain_put);
  Nbre=1000;
  for j=1:Nbre do
    Bin=simul_bin(N,p);
    tau(j)=temps_optimal_option(Bin,V,gain_put);
  end

  QUESTION: proba=QUE VAUT CETTE PROBA;
endfunction;

function main_9()
  N = 50;
  for x_0=[60,70,80,100,120] do
     printf("x_0=%f: %f\n",x_0,compute_proba(x_0,N));
  end
endfunction
```

17. Tracer la frontière d'exercice en fonction du temps.

```
function s=frontiere(V)
  // Calcule la frontière d'exercice t -> s(t) t\in[0,N]
  N=size(V);
  N=N(1)-1;
  s=0;
  for n=0:N do
    for j=[0:n] do
      x=x_0*u^j*d^(n-j);
      if V(n+1,j+1) > gain_put(x) then
        s(n+1)=x_0*u^(j-1)*d^(n-j+1);break;
      end;
    end
  end;
endfunction

function main_10()
  N=1000;
  r_0=0.05;r=r_0/N;
  sigma=0.3;
  K=100;x_0=70;

  p=1/2;d=1-sigma/sqrt(N);u=1+sigma/sqrt(N);

  V=prix_am_vect(x_0,N,gain_put);
  front=frontiere(V);
  plot2d(front);
endfunction;
```

2 Un problème de recrutement

On reçoit, consécutivement, N candidats à un poste. Les circonstances m'imposent de décider tout de suite du recrutement (soit on recrute la personne que l'on vient de recevoir, soit on la refuse définitivement). On cherche à maximiser la proabilité de recruter le meilleur candidat. Quelle est la meilleure stratégie ?

On a vu en cours que l'on peut se ramener à $(S_1, \ldots, S_N)$ est une suite de variables aléatoires indépendantes de Bernouilli $1/n$, qui est une chaîne de Markov sur l'espace $E = \{0, 1\}$, non homogène, de matrice de transition, dépendant du temps, P_n

$$\begin{array}{rcl} P_n(0,0) & = & P_n(1,0) = 1 - \frac{1}{n+1} = \frac{n}{n+1}, \\ P_n(0,1) & = & P_n(1,1) = \frac{1}{n+1}. \end{array}$$

On chercher à maximiser $\mathbf{P}(\tau \text{ est le meilleur}) = \mathbf{E}\left(\frac{\tau}{N} S_\tau\right)$ parmi tous les temps d'arrêt.

1. Ecrire un programme `Scicoslab` qui calcule la solution $(u(n, 0 \text{ ou } 1), 0 \leq n \leq N)$ de l'équation de programmation dynamique donnée par (voir cours)

$$\left\{ \begin{array}{l} u(n,x) = \max\left\{\frac{n}{n+1} u(n+1,0) + \frac{1}{n} u(n+1,1), \frac{n}{N} x\right\}, n < N, \\ u(N,x) = x, \end{array} \right.$$

   On calcule $(u(n, \{0, 1\}), 0 \leq n \leq N)$ à l'aide de l'équation de programmation dynamique. On désigne l'"obstacle" par $(f(n, \{0, 1\}), 0 \leq n \leq N)$. Notez que $f(n, 0) = 0$ et $f(n, 1) = n/N$.

```
  function u=compute_u(N)
    u=zeros(N,2);
    u(N,:)=[0,1];
    for n=[N-1:-1:1] do
      temp = (n/(n+1))*u(n+1,1) + (1/(n+1))*u(n+1,2);
      // temp >= 0, donc u(n,0)=max(temp,0)=temp
      u(n,:)=[temp,max(temp,n/N)];
    end;
  endfunction;
```

2. Tracer les courbes $u(n,0)$, $u(n,1)$ ainsi que "l'obstacle" $\frac{n}{N}$.

```
function main_11()
  N=1000;
  obstacle=[1:N]/N;

  u=compute_u(N);
  plot2d(1:N,obstacle,style=2);
  plot2d(1:N,u(:,2),style=3);// u(n,1)
  plot2d(1:N,u(:,1),style=4);// u(n,0)
endfunction
```

3. Interprétez $u(0,1)$ comme la probabilité de choisir le meilleur candidat avec une stratégie optimale ? Tracer la courbe N donne

$$\mathbf{P}(\text{Le candidat choisi par une stratégie optimale est le meilleur}),$$

pour $N = 10, 25, 50, 100, 250, 500, 1000$. Vérifier numériquement qu'elle converge vers $1/e \approx 37\%$.

```
function main_12()
  // u(1,2) = P(succés pour la stratégie optimale)
  // On vérifie que cette proba tends vers 1/e = 37\%
  i=0;
  courbe=0;
  valeurs=[10,25,50,100,250,500,1000];
  for N=valeurs do
    u=compute_u(N);
    i=i+1;courbe(i)=u(1,2);
  end
  plot2d(valeurs,courbe)
endfunction
```

4. Vérifier que l'on a $u(n,0) > 0$ pour tout $n < N$ et donc, que $0 = f(n,0) \neq u(n,0)$. Par ailleurs, on a bien sûr, $u(N,0) = 0 = f(N,0)$. Comme le temps d'arret optimal τ est donné par

$$\tau = \inf\{n \geq 0, u(n,S_n) = f(n,S_n)\},$$

en déduire qu'il sera toujours postérieur à τ_{min}, le premier temps où $u(n,1) = f(n,1) = n/N$, (puisque $u(n,0) > 0 = f(n,0)$, sauf lorsque $n = N$) et que c'est la premier instant après τ_{min} pour lequel $S_n = 1$, si $n < N$.
Il découle de ce qui précéde qu'un temps d'arrêt optimal est donné par le premier instant après n_{min} (n_{min} compris) où $S_n = 1$ (ou encore $R_n = 1$, c'est à dire le premier instant où le nouvel arrivant est meilleur que ceux qui l'on précédé).
Ecrire un algorithme de calcul de ce temps τ_{min}.

```
function res=temps_min(N)
 // Calcule le
 // premier temps où u(n,1) = f(n,1) = n/N
 // à ϵ près.
  epsilon= 10*%eps;
  u=compute_u(N);
  for n=[1:N] do
    if (abs(u(n,2) - (n/N)) < epsilon) then
         break;
    end
  end
  res=n;
endfunction
```

Vérifier par simulation que le temps (déterministe) τ_{min} "vaut environ" N/e pour N grand. On peut le démontrer sans trop de difficulté à l'aide de la série harmonique.

```
function main_13()
  // Vérification que temps_min vaut environ N/e.
  Taille=1000;
  x=0;
  for N=[1:Taille] do
    x(N)=temps_min(N)/N;
  end
  x=x-1/exp(1);
  nb=size(x,'*');
  plot2d(1:nb,x);
endfunction
```

5. Ecrire une fonction qui calcule la suite des rangs d'insertion (R dans le cours) d'une permutation et une fonction qui calcule la permutation définie par ses rangs d'insertion successifs.

```
function [R] = Omega2R(omega)
  // Calcule les rangs d'insertion R pour un ω donné
  R=zeros(1,length(omega));
  for n=[1:length(omega)] do
    // classe le vecteur omega(1:n) en croissant
    y=gsort(omega(1:n),'g','i');
    // calcule l'indice de omega(n) dans le tableau classe y
    // c'est à dire son classement parmis les n premiers
    R(n)=find(omega(n)==y);
```

```
  end
endfunction

function [omega] = R2Omega(R)
// Calcule ω connaissant les rangs d'insertion

  N=length(R);
  temp=zeros(1,N);
  for n=[1:N] do
     // On place le n-ième individu à sa place: la R(n)-ième
     temp=[temp(1:R(n)-1),n,temp(R(n):n-1)];
  end;
  // On obtient une application  temp qui au rang associe l'individu.
  // Mais  ω, c'est l'application qui à l'individu asscocie son rang:
  // c'est l'inverse de  temp qui se calcule par
  for  n=[1:N] do omega(temp(n))=n;end;
  // ou si l'on est à l'aise en Scilab:
  // omega(temp)=1:N;//!!
endfunction

function main_14()
  // test sur une permutation
  N=10;
  omega=grand(1,'prm',[1:N]')';// tirage d'une permutation aléatoire.
  R=Omega2R(omega);
  // Retrouve t'on omega ?
  and(R2Omega(R)==omega)
endfunction
```

6. Vérifier par simulation que la probabilité d'obtenir le meilleur candidat, lorsque l'on utilise la stratégie optimale, est de l'ordre de $1/e \approx 37\%$. Ce qui n'est pas génial, mais c'est le mieux que l'on puisse faire (sans connaitre l'avenir !).

```
function n=temps_optimal(omega,tau_min)
// Calcule le temps d'arret optimum
// pour la permutation ω

  // Calcule la suite S à partir de omega (omega->R->S)
  S=(Omega2R(omega)==1);

  // Le temps optimal se situe après τmin et c'est le premier instant
  // où S(n)=1 apres ce temps, sauf si n=N, auquel cas on est oblige
  // de prendre  le dernier candidat.
  for n=[tau_min:N] do
    QUESTION: if CONDITION D'ARRET OPTIMAL then break; end;
  end
  // Si on sort de cette boucle avec n=N et N est bien optimal.
endfunction
```

On teste dans un cas particulier.

```
function main_15()
  // Verification que la probabilité d'obtenir
  // le meilleur est de l'ordre de 1/e
  N=100;
  Taille=10000;
  tau_min=temps_min(N);

  ss=0;
  for i=[1:Taille] do
    omega=grand(1,'prm',[1:N]');
    tirages(i)=temps_optimal(omega,tau_min);

    // tirages(i) est il le meilleur ?
    if (omega(tirages(i))==1) then  ss=ss+1;end;
  end
  proba=ss/Taille;
  printf("probabilité d''obtenir le meilleur: %.3f\n",proba);

  // histo_discret(tirages,N);
endfunction
```

Décisions dans l'incertain

Le problème du vendeur de journaux

Jean-Philippe CHANCELIER

Mai 2020

LE PROBLÈME DU VENDEUR DE JOURNAUX (UNE PÉRIODE)

- Chaque matin, le vendeur doit décider d'un nombre de journaux à commander $u \in \mathbb{U} = \{0, 1, \ldots\}$ au prix unitaire $c > 0$.
- La demande du jour est incertaine $w \in \mathbb{W} = \{0, 1, \ldots\}$
  - Si à la fin de la journée il lui reste des invendus : coût unitaire $c_S \in \mathbb{R}$.

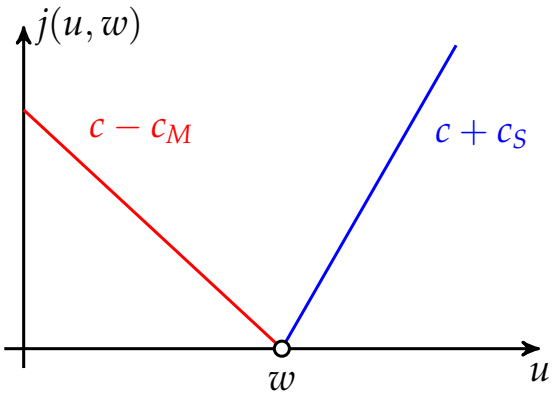
$$c_S(u - w)_+ = c_s \underbrace{\max(u - w, 0)}_{\text{invendus}} \quad c > -c_S$$

  - Si à la fin de la journée il n'a pas pu faire face à la demande on associe un coût unitaire c_M. Le coût lié à la non satisfaction de la demande est

$$c_M(w - u)_+ = c_M \underbrace{\max(w - u, 0)}_{\text{manquants}}$$

FACE À UNE DEMANDE ALÉATOIRE, LE COÛT EST ALÉATOIRE

$$j(u,w) = \underbrace{cu}_{\text{achats}} + \underbrace{c_s(u-w)_+}_{\text{invendus}} + \underbrace{c_M(w-u)_+}_{\text{manquants}}$$



$\text{Argmin}_{u\in\mathbb{U}} j(u,w) = \{w\}$: Quantité inconnue w !

CHOIX D'UN CRITÈRE EN ESPÉRANCE

Le choix du critère : Attitude face au risque du décideur. Le vendeur de journaux va vendre ses journaux chaque jours, on choisit ici un critère en espérance.

$$\min_{u\in\mathbb{U}} J(u) \quad \text{avec } J(u) := \underbrace{\mathbb{E}_w[j(u,w)]}_{\text{Espérance}}$$

Ce n'est pas un critère unique : Un vendeur de journaux pessimiste pourrait minimiser le pire des cas (utilise le support de la loi de la demande) :

$$\min_{u\in\mathbb{U}} J(u) \quad \text{avec } J(u) := \underbrace{\max_{w\in\overline{\mathbb{W}}} j(u,w)}_{\text{Pire des cas}}$$

MINIMISATION DU COÛT EN ESPÉRANCE

- On suppose que la demande, W, est une variable aléatoire et le vendeur de journaux connaît, distribution $\mathbb{P}_W$, la loi de W.
- Le coût à minimiser devient

$$J(u) = \mathbb{E}_W[cu + c_s(u - W)_+ + c_M(W - u)_+]$$

- Le problème du vendeur de journaux devient

$$u^\star \overset{?}{\in} \operatorname*{Argmin}_{u \in \mathbb{U}} J(u)\,.$$

Le vendeur de journaux prends une décision déterministe face à un aléa dont il connaît la loi mais pas la réalisation (Décision-Hasard).

MINIMISATION DU COÛT EN ESPÉRANCE

Commande optimale $u^\star$:

- Si $c_M \leq c$ alors J est une fonction croissante et ne rien commander est optimal.
- Si $c_M > c$ alors J atteint son minimum en $u^\star$:

$$u^\star = \inf\{z \in \mathbb{R} \,|\, F(z) \geq (c_M - c)/(c_M + c_S)\}$$

où F est la fonction de répartition de W, $F(z) = \mathbb{P}(W \leq z)$.

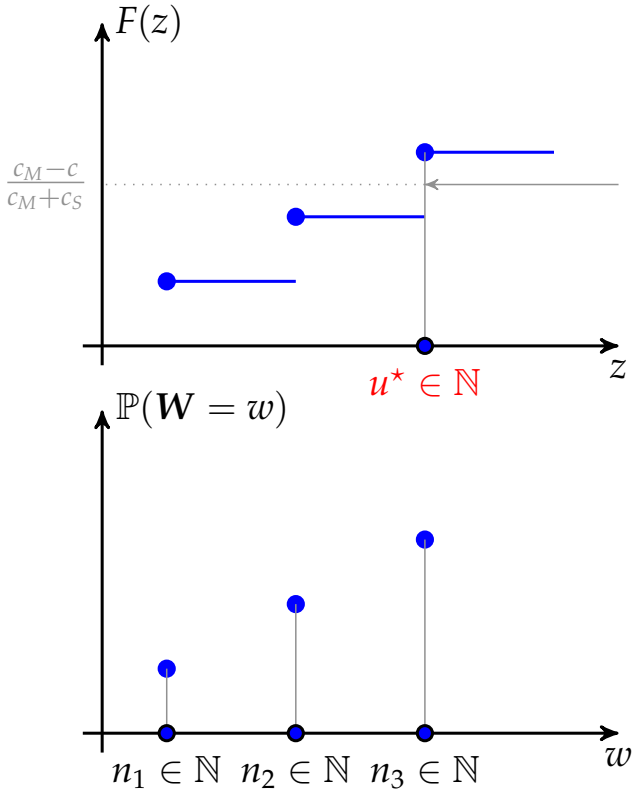
- Si W est une variable aléatoire entière alors $u^\star \in \mathbb{N}$

En effet :

$$\begin{aligned} J(u) &= c_M \mathbb{E}_W[W] + (c - c_M)u + (c_M + c_S)\mathbb{E}_W[(u - W)_+] \\ &= c_M \mathbb{E}_W[W] + (c - c_M)u + (c_M + c_S)\int_0^u F(z)dz \end{aligned}$$

J est continue et coercive. Pour $c_M > c$ elle décroît puis croît avec un minimum en $u^\star$.

MINIMISATION DU COÛT EN ESPÉRANCE

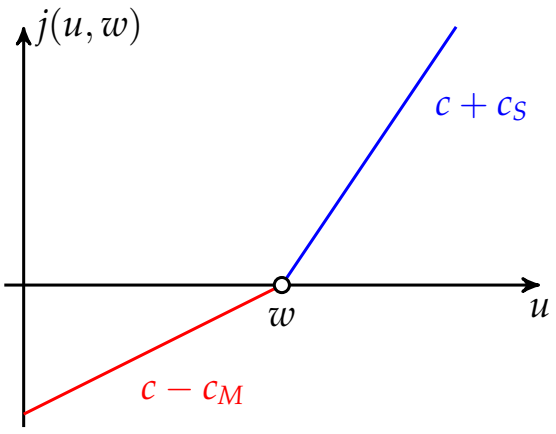


La commande optimale prend des valeurs entières

$$u^\star = \inf \left\{ z \in \mathbb{R} \,|\, F(z) \geq \frac{c_M - c}{c_M + c_S} \right\},$$

si la demande est une variable aléatoire entière.

MINIMISATION DU COÛT EN ESPÉRANCE $c_M \leq c$

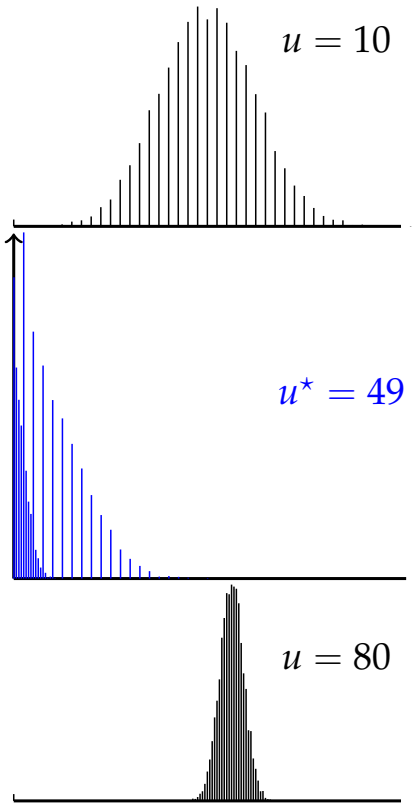


On remarque que

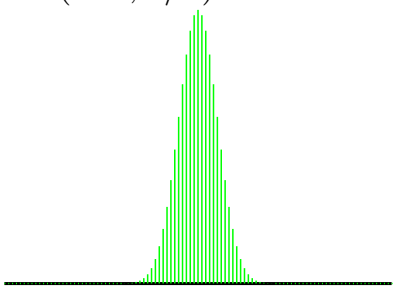
$$\min_{u \in \mathbb{U}} J(u) \geq \min_{u = \phi(\boldsymbol{W})} J(u) = \mathbb{E}_{\boldsymbol{W}}[\min_{u \in \mathbb{U}} j(u, \boldsymbol{W})]$$

Or $\min_{u \in \mathbb{U}} j(u, w)$ est atteint pour $u = 0$ pour tout w et donne une commande admissible pour le problème $\min_{u \in \mathbb{U}} j(u, w)$.

Distribution des coûts $j(u, \boldsymbol{W})$



Distribution de la demande bin$(100, 1/2)$.



Stock initial et coût fixe d'approvisionnement

On suppose ici que l'on peut avoir un stock initial $x \in \mathbb{Z}$ et que toute commande à un coup fixe c_F. Le coût à minimiser devient maintenant :

$$\begin{aligned}\widetilde{J}(u) &= \mathbb{E}_{\boldsymbol{W}}[c_F \mathbb{I}_{\{u>0\}} + cu + c_s(x+u-\boldsymbol{W})_+ + c_M(\boldsymbol{W}-x-u)_+] \\ &= c_F \mathbb{I}_{\{u>0\}} - cx + \mathbb{E}_{\boldsymbol{W}}[j(u+x, \boldsymbol{W})] \\ &= c_F \mathbb{I}_{\{u>0\}} + J(u+x) - cx\end{aligned}$$

Commande optimale $u^\star(x)$ pour $c_M > c$:

La commande optimale est une stratégie (s, S) :

$$u^\star(x) = (S-x)\mathbb{I}_{\{x \le s\}}$$

Quand le stock devient en dessous de s, on commande des journaux de façon à ramener le stock à S. La valeur de S est donnée $\text{Argmin}_{u \in \mathbb{U}} J(u)$.

Stock initial et coût fixe d'approvisionnement

Soit S le stock donné par $\{S\} = \text{Argmin}_{u\in\mathbb{U}} J(u)$.

- Si $x \geq S$, Il est optimal de ne rien faire ($J(\cdot) \nearrow$ et $c_F \geq 0$) :

$$\widetilde{J}(0) = J(x) - cx \leq J(x+u) - cx + c_F = \widetilde{J}(u)$$

- Si $x \leq S$, comme $J(\cdot)$ est minimale en S :
  - Si on commande, il faut commander $u = S - x$ de coût

$$\widetilde{J}(u) = J(S) - cx + c_F$$

  - Si on ne commande rien le coût est $J(x) - cx$

On doit choisir la solution la moins couteuse, Il faut donc compléter son stock jusqu'à S si $J(x) \geq c_F + J(S)$ et ne rien faire sinon. De plus, on remarque que :

$$\{x \,|\, J(x) \geq c_F + J(S)\} = \{x \,|\, x \leq s\}$$

où s est donné par

$$\text{où} \quad s := \sup\{z \in (-\infty, S) \,|\, J(z) \geq c_F + J(S)\}$$

Stock initial et coût fixe d'approvisionnement

Posons $\boldsymbol{X}_0 = x$ et $\boldsymbol{X}_1 = f(\boldsymbol{X}_0, u)$ avec $f(x,u) := x + u - w$ et $\widetilde{j}(u, x_1) := c_F \mathbb{I}_{\{u>0\}} + cu + c_s(x_1)_+ + c_M(-x_1)_+$

Le vendeur de journaux : problème dynamique

$$\min_{U\in\mathcal{U}, X_1, X_0} \mathbb{E}_W[\widetilde{j}(u, \boldsymbol{X}_1)]$$

sous contraintes

$$\boldsymbol{X}_0 = x \quad \boldsymbol{X}_1 = f(X_0, u, \boldsymbol{W})$$

Avec une contrainte de non-anticipativité

$$\mathcal{U} = \{U : \mathbb{X}_0 \to \mathbb{N}\}$$

La commande du nombre de journaux est contrainte à n'utiliser que l'information disponible $\boldsymbol{X}_0 = x$ et pas la demande future $\boldsymbol{W}$.

UNE "CHAÎNE" DE MARKOV

Les deux stocks de journaux $\boldsymbol{X}_0$ et $\boldsymbol{X}_1$ peuvent être vus comme les états d'une chaîne de Markov controlée. On peut autoriser des stocks négatifs et on a alors a priori $X_0 \in \mathbb{Z}$ et $X_1 \in \mathbb{Z}$.

- Supposons $u \in \mathbb{N}$ fixé, la matrice de transition de la chaîne de Markov est

$$P^u_{x_0,x_1} = \mathbb{P}(\boldsymbol{X}_1 = x_1 | \boldsymbol{X}_0 = x_0)$$

$$P^u_{x_0,x_1} = \begin{cases} \mathbb{P}(\boldsymbol{W} = w_0) & \text{si } x_1 = x_0 + u - w_0 \\ 0 & \text{sinon.} \end{cases}$$

- Supposons que u puisse être choisi comme une fonction de X_0, $u = \phi(\boldsymbol{X}_0)$ alors

$$P^\phi_{x_0,x_1} = \begin{cases} \mathbb{P}(\boldsymbol{W} = w_0) & \text{si } x_1 = x_0 + \phi(x_0) - w_0 \\ 0 & \text{sinon.} \end{cases}$$

- Étudier des problèmes sur N pas de temps sera l'objet du prochain cours.

TEMPS D'ARRÊT ET PROBLÈME DE CONTÔLE

Soit P la matrice de transition d'une chaine de Markov d'espace d'état $\mathbb{X}$. On considère $\mathbb{Y} = \mathbb{X} \cup \Delta$ et deux matrices de transitions

$$\forall (x,y) \in \mathbb{X} \quad P^C_{x,y} = P_{x,y} \quad \text{et} \quad P^C_{\Delta,\Delta} = 1$$

et

$$\forall (x,y) \in \mathbb{X} \quad P^S_{x,y} = 0 \quad \text{et} \forall x \in \mathbb{Y} \quad P^S_{x,\Delta} = 1$$

On cherche à évaluer un coût sur un pas de temps pour la chaîne partant de $X_0 = x \in \mathbb{X}$:

$$v(x) = \sup_{u \in \{C,S\}, X_0 = x} \mathbb{E}\left[c(u, X_0) + f_1(X_1)\right]$$

où la matrice de transition de la chaîne est P^u si le contrôle utilisé est u et avec $c(C,\cdot) = 0$ et $c(S,x) = f_0(x)$ et $f_1(\Delta) = 0$. On obtient pour $x \in \mathbb{X}$

$$v(x) = \sup \left(\sum_{y \in \mathbb{X}} P_{x,y} f_1(y), f_0(x) \right)$$

UN PROGRAMME LINÉAIRE

Le problème du vendeur de journaux

$$\min_{u \in \mathbb{U}} J(u) = \mathbb{E}_{\boldsymbol{W}}[j(u, \boldsymbol{W})]$$

avec

$$j(u, w) = c_M w + (c - c_M)u + (c_M + c_S)(u - w)_+$$

peut s'écrire comme un problème linéaire

EXPLICITONS L'ESPÉRANCE

La demande $\boldsymbol{W}$ prends un nombre fini S de valeurs w_s, $s \in \mathbb{S}$ avec probabilité π_s

$$\mathbb{P}(\boldsymbol{W} = s) = \pi_s \, \forall s \in \mathbb{S}$$

$$\sum_{s \in \mathbb{S}} \pi_s = 1 \text{ and } \pi_s \geq 0 \ , \quad \forall s \in \mathbb{S}$$

Le critère à minimiser s'écrit alors

$$J(u) = \sum_{s \in \mathbb{S}} \pi_s \left(c_M w_s + (c - c_M)u + (c_M + c_S)(u - w_s)_+ \right)$$

Explicitons le max $(\cdot)_+$ avec des variables supplémentaires (Optim S1)

$$
\begin{aligned}
J(u) &= \sum_{s\in\mathbb{S}} \pi_s \left(c_M w_s + (c - c_M)u + (c_M + c_S)(u - w_s)_+\right) \\
&= \sum_{s\in\mathbb{S}} \pi_s \min_{\substack{v_s\geq 0 \forall s\in\mathbb{S} \\ v_s \geq u - w_s \ \forall s\in\mathbb{S}}} c_M w_s + (c - c_M)u + (c_M + c_S)v_s \\
&= \min_{\substack{v_s\geq 0 \forall s\in\mathbb{S} \\ v_s \geq u - w_s \ \forall s\in\mathbb{S}}} \sum_{s\in\mathbb{S}} \pi_s \left(c_M w_s + (c - c_M)u + (c_M + c_S)v_s\right)
\end{aligned}
$$

Le problème du vendeur de journaux comme un programme linéaire

$$
\min_{u\in\mathbb{U},(v_s)_{s\in\mathbb{S}}\in\mathbb{V}^{\mathbb{S}}} \sum_{s\in\mathbb{S}} \pi_s c_M w_s + (c - c_M)u + (c_M + c_S)v_s
$$

sous les contraintes

$$
\begin{aligned}
&v_s \geq u - w_s \quad \forall s \in \mathbb{S} \\
&v_s \geq 0 \quad \forall s \in \mathbb{S} \\
&u \geq 0
\end{aligned}
$$

On obtient un programme linéaire en nombres entiers ($\mathbb{U} = \mathbb{N}$ et $\mathbb{V} = \mathbb{N}$) et on peut étudier la version relâchée ($\mathbb{U} = \mathbb{R}$ et $\mathbb{V} = \mathbb{R}$).

Question :

Est-ce que la version relâchée donne une solution entière ?

LA CONTRAINTE DE MESURABILITÉ

Le fait que la commande $u \in \mathbb{U}$ doit être la même pour toutes les réalisations de w_s peut s'exprimer en introduisant de nouvelles commandes $u_s \in \mathbb{U}^{\mathbb{S}}$ (duplication de variables) et en les contraignant a être toutes égales, $u_s = u \quad \forall s \in \mathbb{S}$.

$$\min_{u \in \mathbb{U}, u_s \in \mathbb{U}^{\mathbb{S}}, (v_s)_{s\in\mathbb{S}} \in \mathbb{V}^{\mathbb{S}}} \sum_{s\in\mathbb{S}} \pi_s c_M w_s + (c - c_M)u_s + (c_M + c_S)v_s$$

sous les contraintes

$$v_s \geq u - w_s \quad \forall s \in \mathbb{S}$$

$$v_s \geq 0 \quad \forall s \in \mathbb{S}$$

$$u \geq 0$$

$$u_s = u \quad \forall s \in \mathbb{S}$$

Si on dualise les contraintes $u_s = u \quad \forall s \in \mathbb{S}$ on découple les problèmes s par s. Des méthodes numériques sont basées sur cette idée.

POUR ALLER PLUS LOIN : COURS DE L'ÉCOLE DES PONTS

- 1A : cours d'« Optimisation » (matrices unimodulaires)
- 2A :
  - cours de « Recherche Opérationnelle »,
  - cours d'« Optimisation et contrôle »,
  - cours « Modéliser l'Aléa »

Cours de Décision dans l'incertain
Exercices : 24 avril 2020.

Exercice 1 1. On note F, la fonction de répartition d'une variable aléatoire réelle W, i.e. $F(z) = \mathbb{P}(W \leq z)$. Montrer que F est (toujours !) continue à droite. A quelle condition F est elle continue en un point z ?

2. Montrer que si u et w sont deux nombres réels positifs, on a $\int_0^u \mathbf{1}_{\{w \leq z\}} dz = (u - w)_+$. En déduire que, si W est une variable aléatoire positive :

$$\mathbb{E}\left((u - W)_+\right) = \int_0^u F(z)dz,$$

où $F(z) = \mathbb{P}(W \leq z)$ est la fonction de répartition de W.

Exercice 2 On suppose que c_M, c et c_S sont des nombres réels positifs et W une variable aléatoire positive. On définit, comme dans le cours, la perte moyenne résultant d'une commande en quantité u positive

$$J(u) = \mathbb{E}\left[c_F \mathbf{1}_{\{u>0\}} + cu + c_S(u - W)_+ + c_M(W - u)_+\right].$$

1. Vérifier que $J(u) = c_M\mathbb{E}(W) + (c - c_M)u + (c_M + c_S)\mathbb{E}\left((u - W)_+\right)$.
2. Montrer que $J(u) = c_M\mathbb{E}(W) + (c - c_M)u + (c_M + c_S)\int_0^u F(z)dz$ et que J est continue et admet une dérivée à droite en tout point.
3. On pose
$$u^* = \inf\left\{z \in \mathbb{R}^+, F(z) \geq \frac{c_M - c}{c_M + c_S}\right\}.$$
Montrez que, si W ne prend que des valeurs entières, u^* prend lui aussi des valeurs entières.
4. Montrez que, si $c_M > c$, alors $u^* > 0$ et J est décroissante lorsque $u \in [0, u^*]$ et croissante si $u \geq u^*$ (elle atteint donc son minimum en u^*).
5. Montrer que si $c_M \leq c$ alors $u^* = 0$ et J est une fonction croissante sur $\mathbb{R}^+$ (elle atteint donc son minimum en 0).

Exercice 3 On s'intéresse maintenant à la fonction (dépendant de x) $\tilde{J}^x$, représentant la perte moyenne lorsque le stock initial est égal à x, définie par

$$\tilde{J}^x(u) = \mathbb{E}\left[c_F \mathbf{1}_{\{u>0\}} + cu + c_S(x + u - W)_+ + c_M(W - x - u)_+\right].$$

1. Vérifier que $\tilde{J}^x(u) = c_F \mathbf{1}_{\{u>0\}} + J(u + x) - cx$.

On suppose dans la suite que $c_M > c$. On note S pour l'unique point qui réalise $\text{argmin}_{u \geq 0} J(u)$.

2. Lorsque $x \geq S$, vérifier que pour tout $u > 0$ $\tilde{J}^x(0) \leq \tilde{J}^x(u)$. *En déduire que'il est optimal de ne rien commander.*
3. Lorsque $x \leq S$ et $J(x) < c_F + J(S)$, vérifier que pour tout $u > 0$ $\tilde{J}^x(0) \leq \tilde{J}^x(u)$. *En déduire qu'il est, là aussi, optimal de ne rien commander.*
4. Lorsque $x \leq S$ et $J(x) > c_F + J(S)$, vérifier que pour tout $u > 0$ $\tilde{J}^x(u) \leq \tilde{J}^x(S)$. *En déduire qu'il est optimal de commander la quantité $S - x$.*

5. On définit s par (notez que s est forcément inférieur à S)

$$s = \sup \{z \leq S, J(z) \geq c_F + J(S)\}.$$

Vérifier que $J(x) \geq c_F + J(S)$ est équivalent à $x \leq s$. En déduire que la commande optimale $u^*(x)$ partant d'un stock initial x est donnée par

$$u^*(x) = (S - x)\mathbf{1}_{\{x \leq s\}}.$$

Le vendeur de journaux

Jean-Philippe Chancelier

March 23, 2018

Contents

1 Le problème du vendeur de journaux (à une période de temps)

- Chaque matin, le vendeur doit décider d'un nombre de journaux à commander $u \in \mathbb{U} = \{0, 1, \ldots\}$ au prix unitaire $c > 0$.
- La demande du jour est incertaine $w \in \mathbb{W} = \{0, 1, \ldots\}$
  - Si à la fin de la journée il lui reste des invendus: coût unitaire $c_S \in \mathbb{R}$.

$$c_S(u - w)_+ = c_S \max(u - w, 0) \quad c > -c_S$$

  - Si à la fin de la journée il n'a pas pu faire face à la demande on associe un coût unitaire c_M. Le coût lié à la non satisfaction de la demande est

$$c_M(w - u)_+ = c_M \max(w - u, 0)$$

1.1 La demande aléatoire

On veut faire tourner le même code pour plusieurs lois distinctes qui seront des lois discrètes. On utiliser un `select` en Scicoslab qui permettra de choisir la loi utilisée pour les calculs.

```
// la proba a construire est stockée dans un vecteur ligne
test=1,// pour selectionner la loi à utiliser

select test
  case 1 then
   // On utilise ici une loi binomiale de paramètres (n,p)
   // pdf("bin",x,n,p) = n!/x!(n-x)! p^x (1-p)^n-x
   // titre: chaine de caratères décrivant la loi
   // wi: vecteur contenant les valeurs possibles
   // pi: vecteur contenant les probabilités associées
   // mu: calculer la moyenne de la loi
   N=100000;
   // W: un echantillon de taille N de la loi considérée
   n=100;
   p=0.5;
   titre=sprintf("loi binomiale(%d,%5.2f)",n,p);
   wi=0:n;// les valeurs possibles
   pi=binomial(p,n);// les probabilités associées
   mu=n*p;// moyenne de la binomiale
   mu1=pi*wi';// vérification
   if abs(mu-mu1) > 1.E-8 then pause ;end
   // un echantillong de taille N
   N=100000;
   W=grand(1,N,"bin",n,p);
  case 2 then
   // une loi discrète sur trois valeurs
   // par exemple une loi uniforme sur les trois valeurs
   wi=[30,50,80];
   //pi=[ <<A-FAIRE>> ];
   titre=sprintf("loi discrète sur %d valeurs",size(wi,'*'));
   //mu=  <<A-FAIRE>> ];
   // un echantillong de taille N
   N=100000;
   // <<A-FAIRE>>: expliquer pourquoi avec une matrice de transition
   // dont toutes les lignes sont semblalbles on simule des v.a indépendantes.
   // On va donc utiliser grand avec l'option Markov
   P=[],for i=1:length(pi) do P=[P;pi];end
   // <<A-FAIRE>>: recalculer P de façon vectorielle
```

```
    Y=grand(N,'markov',P,1);
    // Y prends ses valeurs dans [1,length(pi)]
    // Il faut construire un vecteur W tel que
    // W(i) soit 30 si Y(i)==1, 50 si  Y(i)==2, 80 si Y(i)==3
    // code cela de la façon la plus économique possible !
    //W = <<A-FAIRE>> Y
  case 3 then
    // loi de Poisson
    // On choisit une loi de poisson de paramètre mu
    n=100;
    p=0.5;
    mu=n*p;// moyenne de la loi de poisson
    wi=0:n;
    // pi=[ <<A-FAIRE>> ];
    titre=sprintf("loi de Poisson de paramètre %f",mu);
    N=100000;
    // W=grand(<<A-FAIRE>>)
end
```

Question 1 *Pour chacune des lois proposées faites un graphique de la fonction de répartitions de la loi et un graphique de la densité de la loi*

On pourra utiliser le squelette de programme suivant.

```
// graphique de la loi de la demande et de sa
// fonction de répartition

if %t then
  function graphique_loi(titre,valeurs,probas)
    // dessin de la loi discrete et de sa fonction de
    // repartition.
    // la loi discrete se caractérise par le vecteur valeurs
    // et le vecteur probas de même taille.

    xset('window',1);
    xbasc();
    // bornes du graphique: rect=[xmin,ymin,xmax,ymax]
    rect=[0,0,max(valeurs)+10,min(max(probas)+0.2,1)]
    plot2d3(valeurs,probas,rect = rect,style = 2);
    xtitle(titre);
    xset('window',2);
    xbasc();
    // dessin de la fonction de repartition
```

```
    // plot2d2: tracé en escalier
    rect=[0,0,max(valeurs)+10,1.1]
    // le vecteur repart doit contenir la valeur de la fonction
    // de repartition associée aux points valeurs
    // (On pourra utiliser la fonction cumsum).
    //A-FAIRE:  repart=[<<A-FAIRE>>];
    plot2d2([0,valeurs],[0,repart],rect = rect,style = 2)
    // plot2d: tracé des points gauches
    plot2d(valeurs,repart,rect = rect,style = -3)
    xtitle(sprintf('Fonction de repartition de la %s',titre));
  endfunction

  graphique_loi(titre,wi,pi);
  xclick();// il faut cliquer sur le graphique pour continuer
end
```

1.2 On utilise tout d'abord la distribution binomiale

On se place dans le cas `test=1`.

Question 2 *Écrire une fonction Scicoslab qui calcule* `j(u,w)` *puis une fonction qui calcule* $J(u)$. *Faire un graphique avec les valeurs proposées des constantes et calculer le nombre de journaux qu'il faut commander*

```
// parametres de la fonction cout

c=10,cm=20,cs=5;

// la fonction cout j(u,w)

function y=j(u,w)
  //A-FAIRE y= <<A-FAIRE>>
endfunction

// La fonction J(u) pour la loi binomiale en
// utilisant wi et pi

function y=J(u)
  //A-FAIRE y= <<A-FAIRE>>
endfunction

// graphique de la fonction J(u) et recherche du minimum
```

```
if %t then
  U=-10:100;
  //Ju: valeur de J(u) quand U prends les valeurs -10:100
  Ju=[];for u=U do
    //A-FAIRE      <<A-FAIRE>>;
  end
  xset('window',1);xbasc();
  plot2d(U,Ju,style = 2);
  xtitle("La fonction J");
  // recherche du minimum de la fonction J et de son argmin
  // La fonction min permet cela.
  //A-FAIRE [Jmin,kmin]=  <<A-FAIRE>>
  //A-FAIRE uopt= U(<<A-FAIRE>>);
  printf("Nombre optimal de journaux a commander %f\n",uopt);
  printf("Moyenne de la demande %f\n",mu);
  plot2d(uopt,m,style = -4);// dessin
  plot2d3(uopt,m);
  xclick();
end
```

Question 3 *Vérifier sur un graphique que le nombre de journaux optimal à commander s'obtient par la formule :*

$$u^\star = \inf\{z \in \mathbb{R} \,|\, F(z) \geq (c_M - c)/(c_M + c_S)\} \tag{1}$$

```
// graphique de la fonction J(u) et recherche du minimum
// en utilisant la fonction F(z)

if %t then
  xset('window',1);
  xbasc();
  fstar=(cm-c)/(cm+cs);
  plot2d([0,wi],fstar*ones([0,wi]),rect = [0,0,max(wi)+10,1.1],style = 1);
  // La fonction de repartition
  Fv=cumsum(pi)
  plot2d2([0,wi],[0,Fv],rect = [0,0,max(wi)+10,1.1],style = 2);
  // Calculer kopt en utilisant le resultat vu dans les slides.
  //A-FAIRE <<A-FAIRE>>
  kopt=plot2d(wi(kopt),Fv(kopt),rect = [0,0,max(wi)+10,1.1],style = -4);
  xclick()
end
```

1.3 On utilise maintenant une loi discrète à 3 valeurs

Dans le cas précédent, on trouve que le nombre de journaux optimal à commander est très voisin de la moyenne de la demande. On cherche ici à construire un exemple ou les deux nombres seront franchement différents.

Question 4 *Reprendre ce qui précède, en vous plaçant maintenant dans le cas* `test=2` *et chercher à caler des valeurs des probabilités qui permettent d'obtenir le résultat souhaité.*

1.4 La loi du coût

Question 5 *Dans les cas* `test=1` *et* `test=2`*, faites un graphique de la loi du coût pour diverses valeurs de la commande u. On procédera de deux façons différentes*

- *En calculant la loi du coût.*
- *En approchant la loi du coût au moyen de tirages de la demande (loi empirique des coûts).*

```
// graphique de la loi du cout en fonction du paramètre u

if %t then
  function graphique_loi_cout(u,wi,pi)
    xbasc();
    xset('window',1);
    // valeurs possible des couts avec eventuelle répétition
    //A-FAIRE   ji1 = <<A-FAIRE>>
    // on trouve les valeurs non répétées avec la fonction unique
    ji=unique(ji1);
    // on claule les probas associées
    pji=0*ji;
    for i=1:size(ji,'*') do
      I=find(ji1==ji(i));// permet de savoir quels w ont contribué a ji(i)
      //A-FAIRE pji(i)= sum(<<A-FAIRE>>)
    end
    moy=pji*ji';
    vmax=min(max(pji+0.2),1);
    xmax=2000;
    plot2d3(ji,pji,rect = [0,0,xmax,vmax],style = 2);
    plot2d3(moy,vmax,rect = [0,0,xmax,vmax],style = 1);
    xtitle(sprintf('Loi de la fonction cout pour u=%f',u))
  endfunction

  // graphique de la loi du cout pour diverses valeurs de u
```

```
  xset('window',1);
  xbasc();
  for u=[0:10:100] do
    graphique_loi_cout(u,wi,pi);
    xclick();
  end
end

// graphique de la loi du cout suivant le paramètre u
// On dessine la loi empirique obtenue avec des tirages
// montecarlo de la demande (On rappelle que c'est les W
// qu'on aconstruit au début du TP).

if %t then
  function graphique_loi_cout_mc(u,Wmc)
    xbasc();
    xset('window',1);
    // valeurs possible des couts obtenus à partir
    // des tirages Wmc
    //A-FAIRE Jv = <<A-FAIRE>>
    // On veut trouver les valeurs distinctes contenues dans le
    // le vecteur Jv
    ji=unique(Jv);
    // Calculer la proba associée à chacune de ces valeurs distinctes.
    //A-FAIRE pji= <<A-FAIRE>>
    moy=mean(Jv);
    vmax=min(max(pji+0.2),1);
    xmax=2000;
    plot2d3(ji,pji,rect = [0,0,xmax,vmax],style = 2);
    plot2d3(moy,vmax,rect = [0,0,xmax,vmax],style = 1);
    xtitle(sprintf('Loi de la fonction cout pour u=%f',u))
  endfunction

  // graphique d'une approximation Monte-Carlo de la loi du cout
  // pour diverses valeurs de u

  for u=[0:10:100] do
    // Les W sont les tirages Monte-Carlo
    graphique_loi_cout_mc(u,W);
    xclick();
  end
end
```

1.5 Un problème linéaire

Question 6 *En utilisant la présentation faite en cours, construire un problème linéaire dont la solution permet de calculer le nombre de journaux optimal à commander*

```
// On regarde ce problème comme un problème linéaire
// on introduit de nouvelles variables z et w pour linéariser les deux max
// cout = c*u + sum_s cs*pi_s* z_s +cm*y_s*w_s
// q in[10,100]
// z_s >= q - x_s
// w_s >= x_s -q
// var=[q;z;w]

function [A,B,C,lb,ub]=linprog_vendeur(wi,pi)
  m=size(pi,'*');
  // A*var <= B
  //A-FAIRE  A= <<A-FAIRE>>
  //A-FAIRE  B= <<A-FAIRE>>
  // le cout (vecteur colonne)
  //A-FAIRE  C= <<A-FAIRE>>
  // bornes min et max
  lb=[0;zeros(m,1);zeros(m,1)];
  ub=[100;%inf*ones(m,1);%inf*ones(m,1)];
endfunction

if %t then
  [A,B,C,lb,ub]=linprog_vendeur();
  [varopt,lagr,fopt]=linpro(C,A,B,lb,ub);
  usharp=varopt(1);
end
```

1.6 Une stratégie $[s, S]$

On regarde maintenant un cas ou le vendeur à déjà des journaux et ou il paye un coup fixe si il commande des journaux. On cherche à retrouver ici le fait que la stratégie optimale est de la forme $[s, S]$.

Question 7 *On se placera dans le cas* `test=1`*. Calculer le nombre optimal de journaux à commander suivant la valeur du stock initial. Vérifier que la stratégie est bien de la forme $[s, S]$: on remonte le stock au niveau S si il est inférieur à s et on ne fait rien sinon. Calculer s par la formule*

$$s := \sup \{z \in (-\infty, S) \,|\, J(z) \geq c_F + J(S)\}$$

```
cf=200;// coût fixe

function y=Jtilde(u,x)
  y=cf*(u > 0)+J(u+x)-c*x
endfunction

xuopt=[];
// valeur possible pour x
xv=0:1:2*max(wi);
U=0:1:2*max(wi);
for x0=xv do
  //A-FAIRE  Jtildeu= <<A-FAIRE>>// valeurs correspondantes pour Jtilde
  //A-FAIRE  uopt = <<A-FAIRE>>// le u optimal pour x0
  xuopt=[xuopt,uopt];// on garde tous les u optimaux
end

// trouver l'indice ou la quantité commandée devient nulle
//A-FAIRE I=find(<<A-FAIRE>>)
// cela permet de trouver s
//A-FAIRE s=<<A-FAIRE>>

// vérifier que la stratégie optimale est [s,S];

xset('window',1);
xbasc();
plot2d2(xv,xuopt,style = 2);// les couples x, u optimal

// retrouver s avec la formule du cours,
// S=uopt est connu on sait que c'est le uopt qui minimise J
for x=0:2*max(wi) do
  //A-FAIRE if <<A-FAIRE>> then kopt=<<A-FAIRE>>;break;end
end
s=kopt;
```

1.7 Correction

```
// -*- Mode:scilab -*-

// Vendeur de journaux
// cout = c*q +csE[(q-D)^+] + cm*E[(D-q)^+];

// g(q) s'écrit aussi
// g(q) = cm*E[D] + int_0^q ((c-cm) + (cm+cs)F(z)dz
```

```
// On regarde le cout pour diverses lois de probas pour D

// On utilise ici une loi binomiale de paramètres (n,p)
// pdf("bin",x,n,p) = n!/x!(n-x)! p^x (1-p)^n-x

test=2;

select test
  case 1 then
   // On utilise ici une loi binomiale de paramètres (n,p)
   // pdf("bin",x,n,p) = n!/x!(n-x)! p^x (1-p)^n-x
   n=100;
   p=0.5;
   titre=sprintf("loi binomiale(%d,%5.2f)",n,p);
   wi=0:n;// les valeurs possibles
   pi=binomial(p,n);// les probabilités associées
   mu=n*p;// moyenne de la binomiale
   mu1=pi*wi';// vérification
   if abs(mu-mu1) > 1.E-8 then pause ;end
   // un echantillong de taille N
   N=100000;
   W=grand(1,N,"bin",n,p);
  case 2 then
   // une loi discrète
   wi=[30,50,80];
   pi=[1/2,1/4,1/4];
   titre=sprintf("loi discrète sur %d valeurs",size(wi,'*'));
   mu=pi*wi';// la moyenne
   // un echantillong de taille N
   N=100000;
   P=ones(size(wi,'*'),1)*pi;
   Y=grand(N,'markov',P,1);
   W=wi(Y);
  case 3 then
   // loi de Poisson
   // On choisit une loi de poisson de paramètre mu
   n=100;
   p=0.5;
   mu=n*p;
   wi=0:n;
   pi=(mu .^wi*exp(-mu)) ./gamma(wi+1);
   //
```

```
   titre=sprintf("loi de Poisson de paramètre %f",mu);
   N=100000;
   W=grand(1,N,'poi',mu);
end

// graphique de la loi de la demande et de sa
// fonction de répartition

if %t then
  function graphique_loi(titre,valeurs,probas)
    // dessin de la loi
    xset('window',1);
    xbasc();
    plot2d3(wi,pi,rect = [0,0,max(valeurs)+10,min(max(probas)+0.2,1)],style = 2);
    xtitle(titre);
    // dessin de la fonction de repartition
    xset('window',2);
    xbasc();
    plot2d2([0,wi],[0,cumsum(pi)],rect = [0,0,max(valeurs)+10,1.1],style = 2)
    plot2d([0,wi],[0,cumsum(pi)],rect = [0,0,max(valeurs)+10,1.1],style = -3)
    xtitle(sprintf('Fonction de repartition de la %s',titre));
  endfunction

  graphique_loi(titre,wi,pi);
  xclick();
end

// paramètres de la fonction cout

c=10,cm=20,cs=5;cf=200;

// la fonction coût j(u,w)

function y=j(u,w)
  y=c*u+cs*max(u-w,0)+cm*max(w-u,0)
endfunction

// La fonction J(u) pour la loi binomiale

function y=J(u)
  y=c*u+cs*pi*max((u-wi'),0)+cm*pi*max((wi'-u),0);
endfunction
```

```
// graphique de la fonction J(u) et recherche du minimum

if %t then
  U=-10:100;
  Ju=[];
  for u=U do Ju=[Ju,J(u)];end
  xset('window',1);
  xbasc();
  plot2d(U,Ju,style = 2);
  xtitle("La fonction J");
  // recherche du minimum
  [m,kmin]=min(Ju);
  uopt=U(kmin);
  printf("Nombre optimal de journaux a commander %f\n",U(kmin));
  printf("Moyenne de la demande %f\n",mu);
  plot2d(U(kmin),m,style = -4);// dessin
  plot2d3(U(kmin),m);
  xclick();
end

// graphique de la fonction J(u) et recherche du minimum
// en utilisant la fonction F(z)

if %t then
  xset('window',1);
  xbasc();
  fstar=(cm-c)/(cm+cs);
  Fv=cumsum(pi);
  plot2d2([0,wi],[0,Fv],rect = [0,0,max(wi)+10,1.1],style = 2);
  plot2d([0,wi],fstar*ones([0,wi]),rect = [0,0,max(wi)+10,1.1],style = 1);
  for k=1:size(Fv,'*') do
    if Fv(k) >= fstar then kopt=k;break;end
  end
  plot2d(wi(kopt),Fv(kopt),rect = [0,0,max(wi)+10,1.1],style = -4);
  xclick()
end

// graphique de la loi du cout en fonction du paramètre u

if %t then
  function graphique_loi_cout(u,wi,pi)
```

```
    xbasc();
    xset('window',1);
    // valeurs possible des couts
    ji1=c*u+cs*max((u-wi),0)+cm*max((wi-u),0);
    ji=unique(ji1);
    pji=0*ji;
    for i=1:size(ji,'*') do
      I=find(ji1==ji(i));
      pji(i)=sum(pi(I));
    end
    moy=pji*ji';
    vmax=min(max(pji+0.2),1);
    xmax=2000;
    plot2d3(ji,pji,rect = [0,0,xmax,vmax],style = 2);
    plot2d3(moy,vmax,rect = [0,0,xmax,vmax],style = 1);
    xtitle(sprintf('Loi de la fonction cout pour u=%f',u))
  endfunction

  // graphique de la loi du cout pour diverses valeurs de u
  xset('window',1);
  xbasc();
  for u=[0:10:100] do
    graphique_loi_cout(u,wi,pi);
    xclick();
  end
end

// graphique de la loi du cout suivant le paramètre u
// On dessine la loi empirique obtenue avec des tirages
// montecarlo de la demande.

if %t then
  function graphique_loi_cout_mc(u)
    xbasc();
    xset('window',1);
    // valeurs possible des couts
    Jv=c*u+cs*max((u-W),0)+cm*max((W-u),0);
    ji=unique(Jv);
    pji=0*ji;
    for i=1:size(ji,'*') do
      I=find(Jv==ji(i));
      pji(i)=size(I,'*')/size(Jv,'*');
```

```
    end
    moy=mean(Jv);
    vmax=min(max(pji+0.2),1);
    xmax=2000;
    plot2d3(ji,pji,rect = [0,0,xmax,vmax],style = 2);
    plot2d3(moy,vmax,rect = [0,0,xmax,vmax],style = 1);
    xtitle(sprintf('Loi de la fonction cout pour u=%f',u))
  endfunction

  // graphique de la loi du cout pour diverses valeurs
  // de u
  for u=[0:10:100] do
    graphique_loi_cout_mc(u);
    xclick();
  end
end

// On regarde ce problème comme un problème linéaire
// on introduit de nouvelles variables z et w pour linéariser les deux max
// cout = c*u + sum_s cs*pi_s* z_s +cm*y_s*w_s
// q in[10,100]
// z_s >= q - x_s
// w_s >= x_s -q

// var=[q;z;w]
// Contraintes inégalité

function [A,B,C,lb,ub]=linprog_vendeur(wi,pi)
  m=size(pi,'*');
  // A*var <= B
  A=[ones(m,1),-eye(m,m),zeros(m,m);-ones(m,1),zeros(m,m),-eye(m,m)];
  B=[wi(:);-wi(:)];
  // le cout
  C=[c;cs*pi(:);cm*pi(:)];
  // bornes min et max
  lb=[0;zeros(m,1);zeros(m,1)];
  ub=[100;%inf*ones(m,1);%inf*ones(m,1)];
endfunction

if %t then
  [A,B,C,lb,ub]=linprog_vendeur();
  [varopt,lagr,fopt]=linpro(C,A,B,lb,ub);
```

```
  usharp=varopt(1);
end

// On regarde maintenant un cas ou le vendeur à déjà des journaux
// et ou il paye un coup fixe si il commande des journaux
// on voit une stratégie [s,S]

function y=Jtilde(u,x)
  y=cf*(u > 0)+J(u+x)-c*x
endfunction

xuopt=[];
xv=0:1:2*max(wi);
for x0=xv do
  U=0:2*max(wi);
  Ju=[];for u=U do Ju=[Ju,Jtilde(u,x0)];end
  g=[];
  [m,kmin]=min(Ju);
  xuopt=[xuopt,U(kmin)];
end

I=find(xuopt==0);
s=xv(I(1)-1);

// vérifier que la stratégie optimale est [s,S];

xset('window',1);
xbasc();
plot2d2(xv,xuopt,style = 2);
plot2d2(xv(1:s),xv(1:s)+xuopt(1:s),style = 1);
xtitle(sprintf("Valeur de s=%d et de S=%d",xv(s),xv(s-1)+xuopt(s-1)));

// trouver s, S=uopt (calculé avant)
x=0:2*max(wi);
Jv=[];
for i=1:size(xv,'*') do Jv=[Jv,J(x(i))];end
costs=Jv-(cf+J(uopt));
I=find(costs <= 0);
if isempty(I) then
  s=0;
else
  s=x(I(1)-1)
```

```
end
```

Décisions dans l'incertain

Le problème du vendeur de journaux en horizon fini

Jean-Philippe CHANCELIER

Mai 2020

RAPPEL SUR LE PROBLÈME À UN PAS DE TEMPS

Le vendeur de journaux : problème dynamique

$$\min_{u \in \mathcal{U}, X_1, X_0} \mathbb{E}_W[\widetilde{j}(u, X_1)]$$

sous contraintes

$$X_0 = x \quad X_1 = f(X_0, u, W_1)$$

avec

$$f(x,u,w) := x + u - w$$
$$\widetilde{j}(u,x) := c_F \mathbb{I}_{\{u>0\}} + cu + \alpha\left(c_s(x)_+ + c_M(-x)_+\right)$$

Le « stock » X_t peut être positif (c'est alors un stock physique) ou négatif (c'est alors moins la quantité de manquants)
La loi de la demande W, est supposée connue (de moyenne finie).
$\alpha \in (0,1]$ est un taux d'actualisation (valoriser des gains futurs à l'instant présent).

LE PROBLÈME À HORIZON T

Le problème sur un horizon fini

$$\min_{\boldsymbol{U}\in\mathcal{U},X} \mathbb{E}_W[\sum_{t=0}^{T-1} \alpha^t \tilde{j}(\boldsymbol{U}_t, \boldsymbol{X}_{t+1})]$$

sous contraintes

$$\boldsymbol{X}_0 = x \quad \boldsymbol{X}_{t+1} = f(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1})$$

- La loi de la demande $\boldsymbol{W}_t$, est supposée connue (de moyenne finie) pour tout t.
- On sera amené à supposer que les demandes $(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_T)$ sont indépendantes.
- $\alpha \in (0, 1]$ est un taux d'actualisation (valoriser des gains futurs à l'instant présent).

LE PROBLÈME À HORIZON T SOUS UNE FORME CANONIQUE

Le problème sur un horizon fini

$$\min_{\boldsymbol{U}\in\mathcal{U},X} \mathbb{E}_W[\sum_{t=0}^{T-1} \alpha^t c_t(\boldsymbol{U}_t, \boldsymbol{X}_t) + \alpha^T K(\boldsymbol{X}_T)]$$

sous contraintes

$$\boldsymbol{X}_0 = x \quad \boldsymbol{X}_{t+1} = f(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1})$$

avec

$$\begin{aligned} c_t(u, x) &:= c_F \mathbb{I}_{\{u>0\}} + cu + c_s(x)_+ + c_M(-x)_+ \\ c_0(u, x) &:= c_F \mathbb{I}_{\{u>0\}} + cu \\ K(x) &:= c_s(x)_+ + c_M(-x)_+ \end{aligned}$$

Calcul du coût

$$
\begin{aligned}
\widetilde{\jmath}(u_0,x_1)+\alpha\widetilde{\jmath}(u_1,x_2) &= \overbrace{c_F\mathbb{I}_{\{u_0>0\}}+cu_0+\alpha\left(c_s(x_1)_+ + c_M(-x_1)_+\right)}^{\widetilde{\jmath}(u_0,x_1)} \\
&\quad + \alpha\overbrace{\left(c_F\mathbb{I}_{\{u_1>0\}}+cu_1+\alpha\left(c_s(x_2)_+ + c_M(-x_2)_+\right)\right)}^{\widetilde{\jmath}(u_1,x_2)} \\
&= \overbrace{c_F\mathbb{I}_{\{u_0>0\}}+cu_0}^{c_0(u_0,x_0)} \\
&\quad + \alpha\overbrace{\left(c_F\mathbb{I}_{\{u_1>0\}}+cu_1+c_s(x_1)_+ + c_M(-x_1)_+\right)}^{c_1(u_1,x_1)} \\
&\quad + \alpha^2\overbrace{\left(c_s(x_2)_+ + c_M(-x_2)_+\right)}^{K(x_2)} \\
&= c_0(u_0,x_0)+\alpha c_1(u_1,x_1)+\alpha^2K(x_2)
\end{aligned}
$$

La contrainte de non anticipativité

- On suppose que le vendeur de journaux chaque jours note la valeur de la demande qui s'est réalisée et conserve cette valeur.
  - À l'instant t, il connaît $(\boldsymbol{W}_1,\cdots,\boldsymbol{W}_t)$ et $\boldsymbol{X}_0$ et peut utiliser cette information pour calculer sa commande de journaux $\boldsymbol{U}_t$.
  - On pourrait donc choisir pour $\mathcal{U}$ les commandes de journaux qui à l'instant t sont fonction de $(\boldsymbol{X}_0,\boldsymbol{W}_1,\cdots,\boldsymbol{W}_t)$.
- Si les bruits sont indépendants et indépendants de $\boldsymbol{X}_0$, on peut montrer que la stratégie optimale pour des fonctions de $(\boldsymbol{X}_0,\boldsymbol{W}_1,\cdots,\boldsymbol{W}_t)$ ne dépend à l'instant t que du stock $\boldsymbol{X}_t$. La forme de la fonction coût à son importance dans ce résultat.
- Noter que dans le cas des chaînes de Markov données par leur matrices de transition le « bruit » n'est pas observé.
- On parle d'observation complète si on observe l'état de la chaîne de Markov.

On commence par un problème avec juste un coût final

Problème $(\mathcal{P}_0)$ démarrant en x à l'instant initial $t = 0$:

$$\begin{aligned} V_0(x) = \min_{X, U \in \mathcal{U}} \quad & \mathbb{E}\left[K\left(X_T\right)\right], \\ \text{s.c.} \quad & X_0 = x \text{ fixé}, \\ & X_{t+1} = f_t\left(X_t, U_t, W_{t+1}\right), \qquad \forall t = 0, \dots, T-1, \end{aligned}$$

- Les bruits $W = (W_t)_{t=1,\dots,T}$.
- Les contrôles $U = (U_t)_{t=0,\dots,T-1}$.
- Les états $(X_t)_{t=0,\dots,T-1}$.

Dynamique Markovienne

Hypothèses Markoviennes : bruits $X_0, W_1, \dots, W_T$ indépendants et les W_t ont même loi.

- Matrice de transition : dynamique non contrôlée $f : \mathbb{X} \times \mathbb{W} \to \mathbb{X}$

$$X_{t+1} = f(X_t, W_{t+1}), \quad P(x, y) = \mathbb{P}\big(f(x, W_1) = y\big)$$

- Matrice de transition : dynamique contrôlée $f : \mathbb{X} \times \mathbb{U} \times \mathbb{W} \to \mathbb{X}$ avec politique Markoviennes $(\phi_s)_{s\in[0,T-1]}$, $U_t = \phi_t(X_t)$

$$X_{t+1} = f(X_t, \phi_t(X_t), W_{t+1}), \quad P_t(x, y) = \mathbb{P}\big(f(x, \phi_t(x), W_1) = y\big)$$

Pour $u \in \mathbb{U}$, soit $P^u := \mathbb{P}\big(f(x, u, W_1) = y\big)$. Pour une politique $(\phi_s)_{s\in[0,T-1]}$ Markoviennes, P_t^ϕ définie par $P_t^\phi(x, y) := P^{\phi_t(x)}(x, y)$

$$X_{t+1} = f(X_t, \phi_t(X_t), W_{t+1}), \quad P_t^\phi(x, y) = P^{\phi_t(x)}(x, y)$$

UNE FAMILLE DE PROBLÈMES

- Problème $(\mathcal{P}_{t_0})$ démarrant en x à t_0 :

$$V_{t_0}(x) = \min_{\boldsymbol{X},\phi(\cdot)} \quad V_{t_0}^{\phi}(x)$$
$$V_{t_0}^{\phi}(x) = \mathbb{E}\left[K\left(\boldsymbol{X}_T\right)\right],$$
$$\text{avec} \quad \boldsymbol{X}_{t_0} = x, \quad \boldsymbol{X}_{t+1} = f_t\left(\boldsymbol{X}_t, \phi_t(\boldsymbol{X}_t), \boldsymbol{W}_{t+1}\right)$$

- Problème $(\mathcal{P}'_{t_0})$ démarrant en μ à t_0 :

$$\mathcal{V}_{t_0}(\mu) = \min_{\mu,\phi(\cdot)} \quad \mathcal{V}_{t_0}^{\phi}(\mu)$$
$$\mathcal{V}_{t_0}^{\phi}(\mu) = \sum_x \mu_T(x)K(x)$$
$$\text{avec} \quad \mu_{t_0} = \mu, \quad \mu_{t+1} = \mu_t P_t^{\phi}$$

- Dynamique $\mu_{t+1}(y) = \sum_x \mu_t(x) P_{x,y}^{\phi(x)}$

LIENS ENTRE LES FONCTIONS VALEURS $\mathcal{V}_{t_0}^{\phi}(\cdot)$ ET $V_{t_0}^{\phi}(\cdot)$

On a :

$$\mathcal{V}_{t_0}^{\phi}(\mu) = \left\langle \mu, V_{t_0}^{\phi} \right\rangle := \sum_x \mu(x) V_{t_0}^{\phi}(x)\,, \text{et} \quad V_{t_0}^{\phi}(x) = \mathcal{V}_{t_0}^{\phi}(\delta_x(\cdot))$$

En effet :

- Problème $(\mathcal{P}_{t_0})$ démarrant en x à t_0 :

$$V_t^{\phi}(x) = (P_t^{\phi} \cdots P_{T-1}^{\phi} K)(x)$$

(Par exemple pour $t = T-1$ $\quad V_{T-1}^{\phi}(x) = \sum_{y \in \mathbb{X}} P_{T-1}^{\phi}(x,y)K(y)$)

- Problème $(\mathcal{P}'_{t_0})$ démarrant en μ à t :

$$\mathcal{V}_t^{\phi}(\mu) = \mu P_t^{\phi} \cdots P_{T-1}^{\phi} K$$

(Par exemple pour $t = T-1$ $\quad \mathcal{V}_{T-1}^{\phi}(\mu) = \sum_{x,y \in \mathbb{X}} \mu(x) P_{T-1}^{\phi}(x,y)K(y)$)

Le cas $T = 1$

$X_0 = x$

$$V(x) = \mathbb{E}\left[K(\boldsymbol{X}_1)\right] = \sum_y P_{x,y} K(y)$$

la loi de $\boldsymbol{X}_0$ est μ

$$\mathcal{V}(\mu) = \mathbb{E}\left[K(\boldsymbol{X}_1)\right] = \sum_x \mu(x) \sum_y P_{x,y} K(y)$$

On obtient

$$\mathcal{V}(\mu) = \sum_y \mu(x) V(x) = \langle \mu, V \rangle \ ,$$

et

$$\mathcal{V}(\delta_{x'}) = \sum_x \delta_{x'}(x) \sum_y P_{x,y} K(y) = \sum_y \mu(x) P_{x',y} K(y) = V(x') \, .$$

Formule Récursive pour $\mathcal{V}_t$

On a la relation suivante :

$$\mathcal{V}_t(\mu) = \min_{\phi_t} \mathcal{V}_{t+1}(\mu P_t^\phi)$$

Preuve : Le problème $(\mathcal{P}'_t)$ démarrant en μ à t :

$$\mathcal{V}_t(\mu) = \min_{\phi(\cdot)} \quad \mu P_t^\phi \cdots P_{T-1}^\phi K$$

- À l'instant t, P_t^ϕ ne dépend que de la fonction ϕ_t.
- Le vecteur ligne $\mu P_{t_0}^\phi$ est à éléments positifs (c'est une loi de probabilité)

$$\mathcal{V}_t(\mu) = \min_{\phi_t} \left\langle \mu P_t^\phi, \min_{(\phi_s)_{s>t}} P_{t+1}^\phi \cdots P_{T-1}^\phi K \right\rangle$$

$$\mathcal{V}_t(\mu) = \min_{\phi_t} \left\langle \mu P_t^\phi, V_{t+1}(\cdot) \right\rangle = \min_{\phi_t} \mathcal{V}_{t+1}(\mu P_t^\phi)$$

Formule Récursive pour V_t avec $t \in \{0, \cdots, T\}$

On a la relation suivante (Équation de Bellman) :

$$V_t(x) = \min_{u \in \mathbb{U}} \mathbb{E}\big(V_{t+1}(f_t(x, u, \boldsymbol{W}_{t+1}))\big)$$
$$u^\sharp(x) \in \operatorname*{Argmin}_{u \in \mathbb{U}} \mathbb{E}\big(V_{t+1}(f_t(x, u, \boldsymbol{W}_{t+1}))\big)$$
$$V_T(x) = K(x)$$

Preuve : On a obtenu pour $\mathcal{V}_t$

$$\mathcal{V}_t(\mu) = \min_{\phi_t} \mathcal{V}_{t_0+1}(\mu P_t^\phi)$$

- $V_t(x) = \mathcal{V}_t(\delta_x(\cdot))$
- $V_{t_0+1}(\delta_x(\cdot)P_t^\phi) = \mathbb{E}\big(V_{t+1}(f_1(x, \phi_t, \boldsymbol{W}_{t+1}))\big)$

Écriture avec la matrice de transition de la chaîne de Markov

Équation de Bellman

$$V_t(x) = \min_{u \in \mathbb{U}} \sum_y P^u_{x,y} V_{t+1}(y)$$
$$V_T(x) = K(x)$$
$$u^\sharp(x) \in \operatorname*{Argmin}_{u \in \mathbb{U}} \sum_y P^u_{x,y} V_{t+1}(y)$$

Preuve :

$$\sum_y P^u_{x,y} V_{t+1}(y) = \sum_y \mathbb{P}\big(f(x, u, \boldsymbol{W}_{t+1}) = y\big) V_{t+1}(y)$$
$$= \mathbb{E}\big(V_{t+1}(f_t(x, u, \boldsymbol{W}_{t+1}))\big)$$

Horizon fini avec coût instantané

Le problème $(\mathcal{P}_0)$ démarrant en x à l'instant initial $t = 0$:

$$\begin{aligned} V_0(x) = \min_{\boldsymbol{X},\boldsymbol{U}\in\mathcal{U}} \quad & \mathbb{E}\left[\sum_{t=0}^{T-1} L_t\left(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\right) + K\left(\boldsymbol{X}_T\right)\right], \\ \text{s.c.} \quad & \boldsymbol{X}_0 = x \text{ fixé}, \\ & \boldsymbol{X}_{t+1} = f_t\left(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\right), \qquad \forall t = 0, \dots, T-1, \end{aligned}$$

Équation de programmation dynamique

$$V_t(x) = \min_{u\in\mathbb{U}} \mathbb{E}\big(L_t(x, u, \boldsymbol{W}_{t+1}) + V_{t+1}(f_t(x, u, \boldsymbol{W}_{t+1}))\big)$$

Preuve

Le problème $(\mathcal{P}_0)$ démarrant en x à $t = 0$ à un coût $\widetilde{V}(0, x)$ où :

$$\begin{aligned} \widetilde{V}_0(z, x) = \min_{\boldsymbol{X},\boldsymbol{U}\in\mathcal{U}} \quad & \mathbb{E}\left[Z_T + K\left(\boldsymbol{X}_T\right)\right], \\ \text{s.c.} \quad & \boldsymbol{X}_0 = x, \boldsymbol{Z}_0 = z, \text{ fixées}, \\ & \boldsymbol{X}_{t+1} = f_t\left(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\right), \\ & \boldsymbol{Z}_{t+1} = \boldsymbol{Z}_t + L_t\left(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\right), \qquad \forall t = 0, \dots, T-1, \end{aligned}$$

Le couple $(\boldsymbol{Z}, \boldsymbol{X})$ est markovien pour des feedbacks : $\phi_t(z, x)$.

Équation de Bellman

$$\begin{aligned} \widetilde{V}_t(z, x) &= \min_{u\in\mathbb{U}} \mathbb{E}\big(\widetilde{V}_{t+1}(z + L_t(x, u, \boldsymbol{W}_{t+1}), f_t(x, u, \boldsymbol{W}_{t+1}))\big) \\ \widetilde{V}_T(z, x) &= z + K(x) \end{aligned}$$

PREUVE

- On montre que $\widetilde{V}_t(z,x) = z + V_t(x)$
- C'est vrai pour $t = T$, en effet $\widetilde{V}_T(z,x) = z + K(x)$. Puis :

$$\widetilde{V}_t(z,x) = \min_{u\in\mathbb{U}} \mathbb{E}\big(z + L_t(x,u,\boldsymbol{W}_{t+1}) + V_{t+1}(f_t(x,u,\boldsymbol{W}_{t+1}))\big)$$
$$\widetilde{V}_t(z,x) = z + \underbrace{\min_{u\in\mathbb{U}} \mathbb{E}\big(L_t(x,u,\boldsymbol{W}_{t+1}) + V_{t+1}(f_t(x,u,\boldsymbol{W}_{t+1}))\big)}_{V_t(x)}$$

- On remarque alors que le min en $u \in \mathbb{U}$ ne dépend que de x. Le contrôle optimal est en feedback seulement sur x.
- On note aussi que $\widetilde{V}_t(0,x) = V_t(x)$, ce qui donne l'équation de Bellman pour le problème avec coût instantané (V).

PLUS COURT CHEMIN DANS UN GRAPHE

Si le chemin $A \to B \to C \to D \to E$ est optimal alors $C \to D \to E$ est optimal.

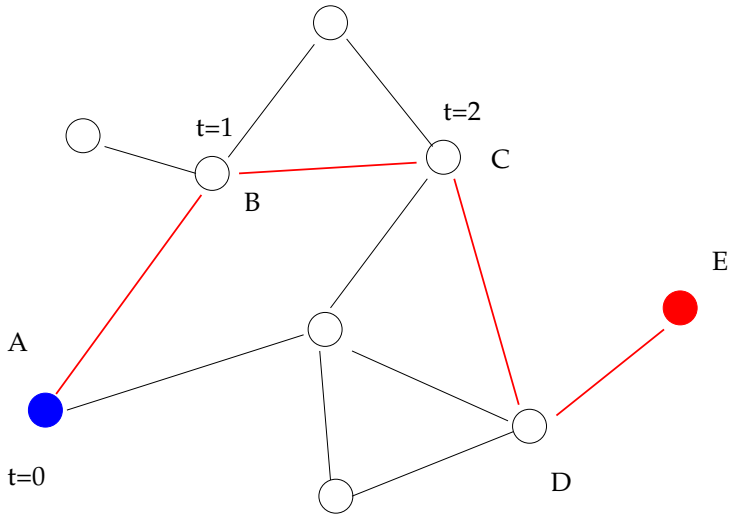


FIGURE – Plus court chemin

PRINCIPE DE PROGRAMMATION DYNAMIQUE

- Plus court chemin pour toutes les paires (i, j) du graphes ($w_{i,j}$ poids de l'arc (i, j)).
- $d_{ij}^{(m)}$: valeur du plus court chemin allant de i à j avec au plus m arcs.
- $d_{i,j}^{(0)} = 0$ si $i = j$ et $+\infty$ sinon.
- Principe de programmation dynamique

$$\begin{aligned} d_{ij}^{(m)} &= \min\left(d_{ij}^{(m-1)}, \min_{1\leq k\leq n, k\neq j}(d_{ik}^{(m-1)} + w_{kj})\right) \\ &= \min_{1\leq k\leq n}(d_{ik}^{(m-1)} + w_{kj}) \end{aligned}$$

- Si pas de cycles de poids négatif alors $d^{(n-1)}$ ou n est le nombre de noeuds du graphe donne la solution du problème.

QUELQUES ÉLÉMENTS TOUJOURS PRÉSENTS

- Un problème de départ : trouver $d^{(n-1)}$ est remplacé par une famille de problèmes $(d^{(m)}, m = 0, \ldots, n-1)$.
- Le problème de départ est bien sûr l'un des problèmes.
- On établit une relation de récurrence entre les $d^{(m)}$.
- Dans la récurrence apparaît un problème d'optimisation local.
- On a remplacé le calcul du plus court chemin par le calcul de la valeur du plus court chemin.
- Le plus court chemin se retrouve en gardant en mémoire l'argmin des problèmes récursifs.

RETOUR SUR LE PLUS COURT CHEMIN

- Fonction valeur :

$$d_{ij}^{(m)} = \min_{1 \leq k \leq n} (d_{ik}^{(m-1)} + w_{kj})$$

- Comment aller de i à j par le plus court chemin. On regarde l'argmin du problème

$$d_{ij}^{(n-2)} = \min_{1 \leq k \leq n} (d_{ik}^{(n-1)} + w_{kj})$$

  Si cet argmin vaut $k^\sharp$ cela veut dire que le plus court chemin $i \rightsquigarrow j = i \rightsquigarrow k^\sharp \to j$
- On regarde alors comment aller de i à $k^\sharp$ en au plus $n-2$ arcs.
- Noter que $d^{(n-1)} = (W)^{(n-1)}$ (dans l'algèbre $(\max, +)$) : carré itérés.

RÉCURSION

- $Fib(n) = (n <= 1)?1 : Fib(n-1) + Fib(n-2)$.

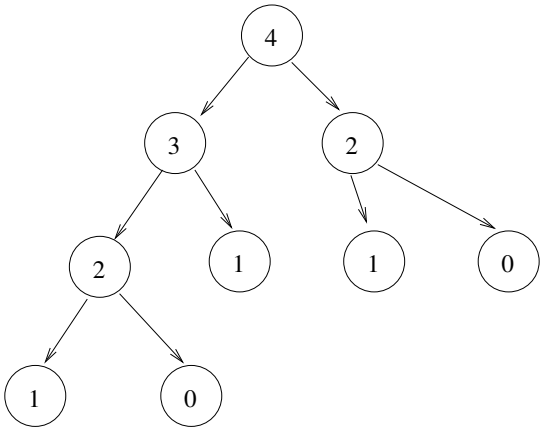


FIGURE – Fibonacci

RÉCURSION

- Complexité exponentielle si l'on y prends garde !
- Solution 1 : partir de $(Fib(0), Fib(1))$ et calculer itérativement (plus de récursion).
- Solution 2 : utiliser la récursion mais garder en mémoire les valeurs déjà calculées (fonction à mémoire ou memoization (en anglais)).

POUR ALLER PLUS LOIN : COURS DE L'ÉCOLE DES PONTS

- cours de « Recherche Opérationnelle »,
- cours d'« Optimisation et contrôle »,
- cours « Modéliser l'Aléa »
- cours « Finance : aspects mathématiques et numériques »

Cours de Décision dans l'incertain
Exercices : xx mai 2020.

Exercice 1 On se donne une suite de fonctions (ou contrôles) $\tilde{u} = (\tilde{u}(n, x_{0:n}), n \geq 0)$ de E^n à valeurs dans un ensemble A ("on choisit la contrôle $\tilde{u}$ en tenant compte de toute la trajectoire avant l'instant n").

$(W_n, n \geq 1)$ une suite i.i.d. à valeurs dans F et x_0 un point de E. Les espaces E, F et A sont supposés finis.

On pose $X_0 = x_0$, puis itérativement on définit $X_{0:n} = (X_0, \ldots, X_n)$, $U_n = \tilde{u}(n, X_{0:n})$ puis X_{n+1} par

$$X_{n+1} = f(X_n, U_n, W_{n+1}). \tag{1}$$

A ce système dynamique contrôlé (1) on associe une famille $(P_u(x, y), x, y \in E, u \in A)$ de matrice de transition, en posant

$$P_u(x, y) = \mathbb{P}(f(x, W_1, u) = y)$$

1. Vérifier que pour un u fixé, $P_u(x, y)$ est une matrice de transition d'un chaine de Markov.
2. Montrer que, pour toute fonction v de E dans $\mathbb{R}$, $\sum_{y \in E} P_u(x, y)v(y) = \mathbb{E}[v(f(x, u, W_1)]$.
3. Montrer que $\mathbb{P}(X_{0:n+1} = x_{0:n+1}) = \mathbb{P}(X_{0:n} = x_{0:n})P_{\tilde{u}(n,x_{0:n})}(x_n, x_{n+1})$.

Exercice 2 On considère l'équation de (Hamilton-Jacobi-)Bellman suivante

$$\begin{cases} v(n, x) = \min_{u \in A} \left\{ \sum_{y \in E} P_u(x, y)v(n + 1, y) + l(n, x, u) \right\}, \ n < N, \\ v(N, x) = h(x). \end{cases} \tag{HJB}$$

Montrer que l'équation précédente définit une *unique* fonction v. Ecrire un algorithme calculant cette fonction v.

Exercice 3 On spécifie le coût d'une stratégie $\tilde{u}$, en suppose que

— le choix de $U_j = \tilde{u}(j, X_{0:j})$ à l'instant j, coûte $l(j, X_j, U_j)$,

— le coût à l'instant final N est donné par $h(X_N)$.

Le coût moyen d'une stratégie $\tilde{u}$ est alors défini par

$$J(\tilde{u}) = \mathbb{E}\left(\sum_{j=0}^{N-1} l(j, X_j, U_j) + h(X_N) \right).$$

On cherche à identifier une stratégie qui minimise $J(\tilde{u})$, parmi tous les contrôles possibles $\tilde{u}$.

Pour cela on considère $\hat{u}(n, x)$ une fonction (pas forcément unique) à valeurs dans A qui réalise le minimum en u présent dans l'équation (HJB) à n et x sont fixés. On note $\hat{X}$ le processus définit par réccurence à partir de $\hat{u}$: $X_0 = x_0$, puis

$$\hat{X}_{n+1} = f(\hat{X}_n, \hat{U}_n, W_{n+1}), \text{ où } \hat{U}_n = \hat{u}(n, \hat{X}_n).$$

Le but est de montrer que $\hat{u}$ réalise le minimum de de J.

1. Vérifier que l'on peut définir une fonction $\hat{u}$ qui réalise le minimum en u présent dans l'équation (HJB) mais que cette fonction n'est pas forcément unique.
2. Montrer, en utilisant la question 3 du premier exercice, que

$$\mathbb{E}\left(v(n+1, X_{n+1})\right) = \sum_{x_{0:n} \in E^{n+1}} \mathbb{P}(X_{0:n} = x_{0:n}) \sum_{x_{n+1} \in E} P_{\tilde{u}(n,x_{0:n})}(x_n, x_{n+1}) v(n+1, x_{n+1}),$$

   puis, en utilisant l'équation (HJB), en déduire que

$$\mathbb{E}(v(n, X_n) - l(n, X_n, U_n)) \leq \mathbb{E}(v(n+1, X_{n+1})). \tag{2}$$

3. En se souvenant que $\hat{u}(n, x)$ réalise le minimum présent dans l'équation (HJB), montrer que

$$\mathbb{E}(v(n, \hat{X}_n) - l(n, \hat{X}_n, \hat{u}(n, \hat{X}_n))) = \mathbb{E}(v(n+1, \hat{X}_{n+1})). \tag{3}$$

4. Montrer par récurrence, en utilisant (2) et $v(N, x) = h(x)$, que :

$$v(0, x_0) \leq \mathbb{E}\left(\sum_{j=0}^{N-1} l(j, X_j, U_j) + h(X_N)\right) = J_{\tilde{u}},$$

5. Montrer à partir de (3) que $v(0, x_0) = \mathbb{E}\left(\sum_{j=0}^{N-1} l(j, \hat{X}_j, \hat{u}(j, \hat{X}_j)) + h(\hat{X}_N)\right) = J_{\hat{u}}$.
6. En déduire l'optimalité de $\hat{u}$: $\min_{\tilde{u}} J(\tilde{u}) = J(\hat{u})$.
   On constate que le contrôle $\hat{u}$ est optimal parmi tous les contrôles mais est de type "feedback" (il ne depend de la trajectoire avant n que par sa dernière valeur X_n).

Le vendeur de journaux sur un horizon T

Jean-Philippe Chancelier

March 23, 2018

Contents

1 Le problème du vendeur de journaux à un pas de temps

On reprend ici le code du T.P précédent et on va utiliser la loi de demande discrète sur trois points que l'on avait considéré sur le problème à un pas de temps.

```
// -*- Mode:scilab -*-

// une loi discrète
wi=[30,50,80];
pi=[1/2,1/4,1/4];
titre=sprintf("loi discrète sur %d valeurs",size(wi,'*'));
mu=pi*wi';// la moyenne
// un echantillong de taille N
N=100000;
P=ones(size(wi,'*'),1)*pi;
Y=grand(N,'markov',P,1);
W=wi(Y);

if %f then
  function graphique_loi(titre,valeurs,probas)
    // dessin de la loi
    xset('window',1);
    xbasc();
    plot2d3(wi,pi,rect = [0,0,max(valeurs)+10,min(max(probas)+0.2,1)],style = 2);
```

```
    xtitle(titre);
    // dessin de la fonction de repartition
    xset('window',2);
    xbasc();
    plot2d2([0,wi],[0,cumsum(pi)],rect = [0,0,max(valeurs)+10,1.1],style = 2)
    plot2d([0,wi],[0,cumsum(pi)],rect = [0,0,max(valeurs)+10,1.1],style = -3)
    xtitle(sprintf('Fonction de repartition de la %s',titre));
  endfunction

  graphique_loi(titre,wi,pi);
  xclick();
end

// paramètres de la fonction cout

c=10,cm=20,cs=5;cf=200;

// la fonction coût j(u,w)

function y=j(u,w)
  y=c*u+cs*max(u-w,0)+cm*max(w-u,0)
endfunction

// La fonction J(u) pour la loi binomiale

function y=J(u)
  y=c*u+cs*pi*max((u-wi'),0)+cm*pi*max((wi'-u),0);
endfunction

// graphique de la fonction J(u) et recherche du minimum

if %t then
  U=-10:100;
  Ju=[];
  for u=U do Ju=[Ju,J(u)];end
  xset('window',1);
  xbasc();
  plot2d(U,Ju,style = 2);
  xtitle("La fonction J");
  // recherche du minimum
  [m,kmin]=min(Ju);
  uopt=U(kmin);
```

```
  S=uopt;
  printf("Nombre optimal de journaux a commander %f\n",U(kmin));
  printf("Moyenne de la demande %f\n",mu);
  plot2d(U(kmin),m,style = -4);// dessin
  plot2d3(U(kmin),m);
  xclick();
end

// On regarde maintenant un cas ou le vendeur à déjà des journaux
// et ou il paye un coup fixe si il commande des journaux
// on voit une stratégie [s,S]

function y=Jtilde(u,x)
  y=cf*(u > 0)+J(u+x)-c*x
endfunction

// On calcule pour x variant de 0 à 2*max(wi)
// la commande optimale de journaux correspondante

xuopt=[];
xv=0:1:2*max(wi);
for x0=xv do
  U=0:2*max(wi);
  Ju=[];for u=U do Ju=[Ju,Jtilde(u,x0)];end
  g=[];
  [m,kmin]=min(Ju);
  xuopt=[xuopt,U(kmin)];
end

I=find(xuopt==0);
s=xv(I(1)-1);

// vérifier que la stratégie optimale est [s,S];

xset('window',1);
xbasc();
plot2d2(xv,xuopt,style = 2);
plot2d2(xv(1:s),xv(1:s)+xuopt(1:s),style = 1);
xtitle(sprintf("Valeur de s=%d et de S=%d",xv(s),xv(s-1)+xuopt(s-1)));

//
```

```
// trouver s, S=uopt (calculé avant)
x=0:2*max(wi);
Jv=[];
for i=1:size(xv,'*') do Jv=[Jv,J(x(i))];end
costs=Jv-(cf+J(uopt));
I=find(costs <= 0);
if isempty(I) then
  s=0;
else
  s=x(I(1)-1)
end

printf("On trouve s=%d et S=%d\n",s,S);
```

Question 1 *Faites éxecuter le code précédent dans scicoslab et vérifiez que la stratégie optimale est bien de la forme* $[s, S]$.

1.1 On cherche maintenant à résoudre le problème du vendeur de journaux sur un horizon T

On propose ici un squelette de programme qui implémente l'équation de Programmation dynamique. Il faut compléter ce programme et résoudre le problème sur un horizon T. À l'instant $t = 1$ le vendeur de journaux fait sa première commande et à l'instant T il ne commande plus. Le cas $T = 2$ correspond donc au problème du vendeur sur une journée.

```
// -*- Mode:scilab -*-
// Vendeur de journaux à T-pas de temps

exec('vendeur-T-un.sci');

alpha=1.0;

function y=ct(u,x)
  // coût instantané pour t autre que la date de départ
  y=cf*(u > 0)+c*u+alpha*cs*max(x,0)+alpha*cm*max(-x,0)
endfunction

function y=c0(u)
  // coût instantané pour t=1
  y=cf*(u > 0)+c*u
endfunction

function y=K(x)
```

```
  // coût final
  y=cs*max(x,0)+cm*max(-x,0)
endfunction

// on transforme la dynamique pour rester dans un domaine
// borné [-100,100]

xmin=-100;xmax=100;

function y=f(x,u,w)
  y=min(max(x+u-w,xmin),xmax)
endfunction

// VT
// On commencera avec T=2 qui doit donner le problème à un pas de temps


T=2;// l'Horizon

etats=xmin:xmax;// les etats possibles dans un domaine borné
allu=0:xmax;// les commandes de journaux possible

// La fonction v_T(x) = K(x)
VT=alpha^T*K(etats);

// On va calculer V_t pour T-1,...,1
// et stocker les résultats dans une liste
// On stocke aussi la commande optimale

L=list(VT);// liste qui permet de stocker (V_T,V_{T-1},.....,V_1)
U=list(0);// liste qui permet de stocker (-- ,U_{T-1},.....,U_1)

// boucle rétrograde en temps
for t=T-1:-1:1 do
  printf("t=%d\n",t);
  Vt=zeros(1,size(etats,'*'));
  Ut=%nan*ones(1,size(etats,'*'));
  Vtp1=L($);
  // On cherche a calculer ici Vt et Ut
  // Vtp1 est la fonction de Bellman au temps t+1
  for i=1:size(etats,'*') do
    x=etats(i);
```

```
    mcost=%inf;
    // boucle sur les commandes possibles
    for k=1:size(allu,'*') do
      u=allu(k);
      // on calcule dans cost le cout associé a la commande u
      // initialisation:
      cost=0;
      // faire une boucle sur les aléas pour calculer
      // E[ ct(u,x) + V_{t+1}(f(x,u,W))]
      for xxx=zzz do
        //AFAIRE ...
        // xtp1: utiliser la dynamique de la chaine
        //AFAIRE xtp1 = ...
        ix=find(xtp1==etats);// donne l'indice de xtp1 dans le
        // vecteurs des etats
        // actualiser le cout
        //AFAIRE cost = cost + ...
      end
      // Il faut maintenant comparer
      // cost au meilleur trouvé jusque la (on minimise)
      // mcost et mettre à jour mcost
      // et kumin (l'indice de la commande qui a donné le meilleur coût)
      //AFAIRE if ... ...  end
    end
    // On stocke pour l'état x d'indice i le coût optimal
    Vt(i)=mcost;
    // et la commande optimale
    Ut(i)=allu(kumin);
  end
  // On range Vt et Ut dans la liste
  L($+1)=Vt;
  U($+1)=Ut;
end

// dessin de la fonction de Bellman au temps t=1;
xset('window',1);
xbasc();
plot2d2(etats,L($))

// dessin de la commande optimale au temps t=1;
xset('window',1);
xbasc();
```

```
plot2d2(etats,U($))
```

Question 2

- *Compléter le code pour résoudre l'équation de Bellman.*
- *Faire tourner le code pour $T = 2$ et retrouver les résultats du problème à un pas de temps.*
- *Faire tourner le code pour $T = 8$ et vérifier que l'on obtient aussi une stratégie $[s, S]$.*
- *Simuler des trajectoires de la chaîne de Markov controlée avec la commande optimale et tracer des trajectoires*
- *Caluler à partir de simulations pour un point inital fixé x un coût moyen et le comparer à la valeur de Bellman calculée.*