Deuxième année F3-F4
Année universitaire 2007/2008

Programmation dynamique

Vincent Barra

Institut Supérieur d'Informatique, de Modélisation et de leurs Applications
Campus des Cézeaux - B.P. 1025 - 63173 AUBIERE CEDEX

Table des matières

La programmation dynamique est née avec la seconde guerre mondiale et les travaux d'Abraham Wald (introduction de l'analyse séquentielle), puis reprise par Richard Bellman, à chaque fois avec des objectifs de résolution de problèmes d'optimisation séquentielle. Ce cours s'articule autour de la formulation de la programmation dynamique proposée par Dimitri Bertsekas en 1995 (D.Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, Athena Scientific, Belmont, Massachussets) : après une introduction du problème par l'exemple, une formulation générale du problème d'optimisation séquentielle est proposée, puis sa résolution par l'algorithme de programmation dynamique est reprise en détail, et illustrée par des exemples.

1 Le problème d'optimisation séquentielle

1.1 Position du problème

Les problèmes d'optimisation séquentielle se proposent, dans une situation donnée, de prendre des décisions de manière séquentielle, les conséquences de chaque décision n'étant pas toujours parfaitement maîtrisées, mais pouvant être anticipées jusqu'à ce que la prochaine décision soit prise. Leur but est de minimiser un coût (ou, de manière équivalente, maximiser un profit), associé à la suite de décisions retenues et à leurs conséquences.
Le plus souvent, les décisions sont fortement dépendantes les unes des autres, et une décision à un moment donné doit non seulement privilégier une baisse du coût local, mais également anticiper les dangers de coûts futurs élevés engendrés par l'orientation qu'elle fait prendre au problème. La programmation dynamique, et l'algorithme qui porte son nom, a pour objectif de trouver le meilleur compromis entre ces objectifs, en ordonnant à chaque étape les décisions selon la somme d'un coût immédiat et de l'espérance des coûts à venir, supposant alors que les décisions futures sont prises de manière optimale
Si le cadre d'application posé est très général, nous n'étudierons ici que les problèmes dans lesquels le nombre de décisions à prendre est fini (on parlera d'**horizon fini**). Le modèle de base se compose alors de deux éléments essentiels :
– un système dynamique à temps discret
– une fonction de coût additive dans le temps

Notons
– N le nombre de périodes considérées,
– $x_k \in S_k$ l'état du systèmé étudié au début de la période $k, k \in \{1 \cdots N\}$ (qui résume les informations sur le passé du système susceptibles d'influencer son évolution future),

– $u_k \in C_k$ la décision (ou **contrôle**) devant être prise à la période k, généralement contrainte à prendre ses valeurs dans un sous-ensemble $U_k(x_k)$ de C_k,
– ω_k un paramètre aléatoire bruitant les données (éventuellement nul), suivant une loi de probabilité $H_k(. \mid x_k, u_k)$.

Le système dynamique qui décrit l'évolution de l'état du processus au cours des N périodes est de la forme :

$$(\forall k \in \{1 \cdots N\}) x_{k+1} = f_k(x_k, u_k, \omega_k)$$

Une fonction de coût $g_k(x_k, u_k, \omega_k)$ est alors associée à chaque période k, qui s'additionnent les unes aux autres pour former le coût total J engendré par le processus complet de décision :

$$J = g_{N+1}(x_k, u_k, \omega_k) + \sum_{k=1}^{N} g_k(x_k, u_k, \omega_k),$$

$g_{N+1}(x_k, u_k, \omega_k)$ étant le coût terminal dépendant de l'état x_{N+1} atteint à la fin du processus.
J est généralement une variable aléatoire (dépendance aux ω_k), et la fonction objectif du problème de minimisation en programmation dynamique est généralement l'espérance de J prise par rapport à la distribution conjointe des variables aléatoires ω_k. La minimisation, quant à elle, est effectuée par rapport aux variables de décision u_k.

1.2 Premières illustrations

1.2.1 Problème de gestion de stocks

On cherche à réapprovisionner un stock en un article pour les N périodes à venir. On suppose donc ici que x_k est le nombre d'unités de l'article dans le stock au début de la période k, u_k est le nombre d'unités commandées et reçues au début de la période k, et que ω_k est la demande des clients pendant la période k. Ces dernières sont supposées être des variables aléatoires indépendantes, de loi de probabilité connues. Si, en cas de rupture de stock, les demandes non satisfaites sont préservées jusqu'à réception du nombre d'articles idoine, le niveau du stock évolue selon le processus dynamique :

$$x_{k+1} = x_k + u_k - \omega_k$$

Les coûts de gestion de ce stock comprennent pour chaque période k :
– un coût de réapprovisionnement, qui si la commande d'un article côute c, est égal à cu_k ;

– un coût $r(x_k, u_k, \omega_k)$ représentant soit un coût de stockage associé aux unités en excès en fin de période, soit à un coût de pénurie encouru pour les demandes n'ayant pu être satisfaites pendant la période.

Ainsi, si r_{N+1} désigne le montant total à payer, l'espérance des coûts totaux est :

$$\mathbb{E}\left(r_{N+1} + \sum_{k=1}^{N} (cu_k + r(x_k, u_k, \omega_k))\right)$$

Pour un stock initial x_1 donné, le problème consiste alors à estimer au mieux les u_k de manière à minimiser cette espérance.

1.2.2 Problème du sac à dos

On considère dans ce problème générique un ensenble de N types d'objets numérotés, chaque objet de type k étant défini par une valeur $c_k \in \mathbb{N}$ et un volume $a_k \in \mathbb{N}$.

Le problème de la sélection d'un sous-ensemble d'objets de valeur maximale ne dépassant pas un volume total V peut être modélisé par le programme linéaire en nombre entiers :

$$\text{maximiser } z = \sum_{k=1}^{N} c_k u_k$$

sous :

$$(\forall k \in \{1 \cdots N\}) u_k \in \mathbb{N}$$

$$\sum_{k=1}^{N} a_k u_k \leq V$$

où u_k est le nombre d'objets de type k effectivement retenu dans le sous-ensemble optimal.

Dans le cadre de la programmation dynamique, ce problème du sac à dos est abordé en N étapes. A l'étape k, l'état x_k modélise l'espace réservé pour les objets de type $j \in \{k \cdots N\}$. On a donc :

$$(\forall k \in \{1 \cdots N\}) x_{k+1} = x_k - a_k u_k$$

La décision u_k consiste quant à elle à choisir le nombre d'objets de type k à inclure dans le sous-ensemble. Bien entendu, u_k doit être positif ou nul et ne doit pas entraîner un dépassement de volume à la fin de l'étape k, d'où $u_k \geq 0$ et $x_k - a_k u_k \geq 0$.

La valeur des objets sélectionnés à l'étape k est $c_k u_k$ et la fonction à maximiser par rapport aux u_k est alors

$$\sum_{k=1}^{N} c_k u_k$$

1.2.3 Analyse des exemples

Bien que modélisés tous deux par la programmation dynamique, ces deux exemples présentent une différence de taille : dans le problème de gestion de stock, la présence des variables aléatoires ω_k implique de déterminer les u_k sans connaître les valeurs exactes des demandes des clients, mais seulement en connaissant leur loi de probabilité. Dans le problème du sac à dos, en revanche, aucun bruit ne vient perturber les données. Cette différence a des répercussions sur l'optimisation du coût total qui peut se faire dans deux contextes différents :

1. toutes les décisions $(u_k)_{k\in\{1\cdots N\}}$ doivent être prises au début du processus de décision, la seule connaissance du système étant son état initial x_1 ;
2. le choix de u_k peut être défini après la prise de connaissance de l'état x_k.

Le second type de scénario fournit bien entendu de meilleurs résultats, et concerne tout particulièrement la programmation dynamique.
Dans ce cadre, la solution du problème d'optimisation séquentielle consiste en une suite de fonctions $u_k = \mu_k(x_k), k \in \{1 \cdots N\}$, définissant pour chaque période k la décision u_k à prendre en fonction de l'état x_k du système qui vient d'être observé. La suite $\pi = (\mu_1 \cdots \mu_N)$ est appelée **politique de décision**, et sera dite **admissible** si à chaque période k et pour chaque état du système correspondant x_k, le décision $u_k = \mu_k(x_k)$ satisfait les contraintes U_k spécifiques du problème.
Pour les **processus de décision stochastiques** (présence de variables ω_k), il s'agit non pas de déterminer les valeurs numériques de u_k mais de spécifier une politique de gestion spécifiant pour chaque période k une quantité optimale u_k.
Pour les **processus de décision déterministes** (variables ω_k absentes ou déterministes), une politique admissible permet de calculer explicitement la suite des décisions $(u_1 \cdots u_N)$, par : $u_k = \mu_k(x_k)$, où les états $(x_j)_{j\in\{2\cdots N\}}$ sont déterminés à partir de l'état x_1 et de la politique retenue par : $x_{k+1} = f_k(x_k, \mu(x_k))$. De plus, le coût associé à la suite de décisions $(u_1 \cdots u_N)$ est le même que celui associé à la politique optimale correspondante, et dans ce cas donc les deux scénarii d'optimisation du coût global sont équivalents.

2 Algorithme de programmation dynamique

2.1 Position du problème

Définition 1. *Une* **politique de décision** *est une suite de fonctions* $(\mu_1 \cdots \mu_N)$ *dans laquelle* μ_k *associe à chaque état* $x_k \in S_k$ *une décision* $u_k = \mu_k(x_k)$.

Définition 2. *Une politique* $\pi = (\mu_1 \cdots \mu_N)$ *est dite* **admissible** *si :*

$$(\forall k \in \{1 \cdots N\})(\forall x_k \in S_k)\ \mu_k(x_k) \in U_k(x_k)$$

L'ensemble des politiques admissibles est noté Π

Etant donné un état initial x_1 et une politique admissible $\pi = (\mu_1 \cdots \mu_N)$, l'évolution du système est régie par le système d'équations :

$$(\forall k \in \{1 \cdots N\}) x_{k+1} = f_k(x_k, \mu_k(x_k), \omega_k)$$

Les fonctions f_k sont les **fonctions de transfert** du système.
Bien souvent, les x_k dépendent des ω_k, et sont donc également des variables aléatoires. L'espérance du coût total associé à x_1 et à π est :

$$J_\pi(x_1) = \mathbb{E}\left(\sum_{k=1}^{N} g_k(x_k, \mu(x_k), \omega_k) + g_{N+1}(x_{N+1})\right) \tag{1}$$

où l'espérance est prise par rapport à la densité conjointe des ω_k.
Le but de l'algorithme de programmation dynamique est de déterminer une politique admissible minimisant (1) (dans le cas où J représente un coût).

Définition 3. *Une politique* π^* *est dite* **optimale** *pour un état initial* x_1 *si elle vérifie :*

$$J^*(x_1) = J_{\pi^*}(x_1) = \min_{\pi \in \Pi} J_\pi(x_1)$$

2.2 Principe d'optimalité de Bellman

Bellman a formulé en 1957 un principe d'optimalité, qui affirme que toute politique optimale est formée de politiques résiduelles optimales, soit à l'aide des notations précédentes :

Théorème 1. *: principe d'optimalité*
Soit $\pi^* = (\mu_1^* \cdots \mu_N^*)$ *une politique optimale pour un problème d'optimisation séquentielle sur* N *périodes. Si, mettant en oeuvre* π^**, l'état* x_j *est atteint à la période j, la politique partielle* $(\mu_j^* \cdots \mu_N^*)$ *est optimale pour le sous-problème sur* $N - j + 1$ *périodes, débutant à l'état j, i.e. minimise :*

$$\mathbb{E}\left(\sum_{k=j}^{N} g_k(x_k, \mu_k(x_k), \omega_k) + g_{N+1}(x_{N+1})\right)$$

Ce théorème suggère en particulier qu'il est possible de construire une politique optimale par morceaux : dans un premier temps, une solution optimale est calculée pour le sous-problème ne faisant intervenir que la dernière étape du problème initial, qui est ensuite utilisée pour déterminer une solution du sous-problème défini par les deux dernières étapes....et le processus est itéré jusqu'à aboutir à x_1. C'est le principe de l'algorithme de programmation dynamique.

2.3 Algorithme de programmation dynamique

Théorème 2. *Soit un problème d'optimisation séquentielle sur N périodes. Pour tout état initial x_1, le coût optimal $J^*(x_1)$ est égal à $J_1(x_1)$, où la fonction J_1 est donnée par la dernière étape de l'algorithme de programmation dynamique, qui procède depuis la période N jusqu'à la période 1, selon les étapes suivantes :*

$$(\forall x_{N+1} \in S_{N+1}) J_{N+1}(x_{N+1}) = g_{N+1}(x_{N+1})$$

$$(\forall k \in \{N \cdots 1\}) J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}\left(g_k(x_k, u_k, \omega_k) + J_{k+1}(f_k(x_k, u_k, \omega_k))\right)$$

où l'espérance à l'étape k est calculée par rapport à la distribution $H_k(. \mid x_k, u_k)$. Enfin, si $\mu_k^*(x_k) = u_k^*$ minimise cette espérance, pour tout x_k et tout k, la politique $\pi^* = (\mu_1^* \cdots \mu_N^*)$ est optimale

Démonstration
Dans la suite, pour simplifier la démonstration, on suppose que les variables aléatoires ω_k sont discrètes.
Pour toute politique admissible $\pi = (\mu_1 \cdots \mu_N)$ et tout $k \in \{1 \cdots N\}$, notons $\pi^k = (\mu_k \cdots \mu_N)$ la restriction de π aux $N - k + 1$ dernières périodes, et $J_k^*(x_k)$ le coût optimal pour ce problème, avec x_k comme état initial. Alors :

$$J_k^*(x_k) = \min_{\pi^k} \mathbb{E}\left(\sum_{j=k}^{N} g_j(x_j, \mu_j(x_j), \omega_j) + g_{N+1}(x_{N+1})\right)$$

Notons enfin $J_{N+1}^*(x_{N+1}) = g_{N+1}(x_{N+1})$
Nous allons montrer par récurrence inverse que les J_k^* sont égales aux fonctions J_k calculées par l'algorithme de programmation dynamique.
– Par définition, $J_{N+1}^* = J_{N+1} = g_{N+1}$ est vrai.
– considérons une étape k et supposons que pour tout état x_{k+1}, $J_{k+1}^*(x_{k+1}) = J_{k+1}(x_{k+1})$.

En remarquant que $\pi^k = (\mu_k, \pi^{k+1})$, il vient :

$$\begin{aligned}
J_k^*(x_k) &= \min_{(\mu_k, \pi^{k+1})} \mathbb{E}_{\omega_k \cdots \omega_N} \\
&\left(g_k(x_k, \mu_k(x_k), \omega_k) + \sum_{j=k+1}^{N} g_j(x_j, \mu_j(x_j), \omega_j) + g_{N+1}(x_{N+1}) \right) \\
&= \min_{\mu_k} \mathbb{E}_{\omega_k} \left[(g_k(x_k, \mu_k(x_k), \omega_k) \right. \\
&\left. + \min_{\pi^{k+1}} \mathbb{E}_{\omega_{k+1} \cdots \omega_N} \left(\sum_{j=k+1}^{N} g_j(x_j, \mu_j(x_j), \omega_j) + g_{N+1}(x_{N+1}) \right) \right] \\
&= \min_{\mu_k} \mathbb{E}_{\omega_k} \left[g_k(x_k, \mu_k(x_k), \omega_k) + J_{k+1}^*(f_k(x_k, \mu_k(x_k), \omega_k) \right] \\
&= \min_{\mu_k} \mathbb{E}_{\omega_k} \left[g_k(x_k, \mu_k(x_k), \omega_k) + J_{k+1}(f_k(x_k, \mu_k(x_k), \omega_k)) \right] \\
&= \min_{u_k \in U_k(x_k)} \mathbb{E}_{\omega_k} \left[g_k(x_k, u_k, \omega_k) + J_{k+1}(f_k(x_k, u_k, \omega_k)) \right] \\
J_k^*(x_k) &= J_k(x_k)
\end{aligned}$$

et la propriété est est vérifiée pour k, ce qui achève la démonstration.

3 Programmes dynamiques déterministes

Dans ce type de problème, il n'existe pas de dépendances à des paramètres aléatoires, les ω_k étant nuls ou prenant des valeurs déterminées.

3.1 Algorithme de programmation dynamique déterministe

Dans les problèmes déterministes, l'évolution du système est complètement déterminée par la donnée de l'état initial x_1 et de la suite des décisions $u_1 \cdots u_N$. La minimisation des coûts totaux sur l'ensemble des politiques admissibles ne procure alors aucun avantage par rapport à celle effectuée sur l'ensemble des suites de décisions. En effet, partant de x_1, le coût atteint par la politique admissible $\pi = (\mu_1 \cdots \mu_N)$ est égal à celui atteint par la suite de décisions $(\mu_1(x_1) \cdots \mu_N(x_N))$, où $x_{k+1} = f_k(x_k, \mu_k(x_k))$. Lors de l'optimisation de

$$\sum_{k=1}^{N} g_k(x_k, u_k) + g_{N+1}(x_{N+1}),$$

il est donc possible de n'envisager le problème que sur l'espace des suites de décisions admissibles, et le programme dynamique déterministe se résoud alors par l'algorithme suivant :

1. Initialisation : $(\forall x_{N+1} \in S_{N+1}) J_{N+1}(x_{N+1}) = g_{N+1}(x_{N+1})$
2. Propagation : $(\forall k \in \{N \cdots 1\})(\forall x_k \in S_k)$ résoudre :

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \{g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k))\}$$

   Stocker $J_k(x_k)$ et $u_k^* = \mu_k^*(x_k)$ solutions de ce problème.
3. Conclusion : pour l'état initial x_1 :
   - le coût optimal est $J_1(x_1)$
   - la suite de décisions optimales est $(\mu_1^*(x_1), \mu_k^*(f_{k-1}(x_{k-1}, \mu_{k-1}(x_{k-1}))),$ $k \in \{2 \cdots N\})$

Exemple : problème d'investissement
Une entreprise souhaite investir ses 500000 euros de bénéfice dans l'amélioration de ses usines pour l'année à venir. Trois sites sont visés, dont les bénéfices annuels espérés (en centaines de milliers d'euros) sont, en fonction de la somme investie pour leur amélioration (en centaines de milliers d'euros) :

Sites	Montants investis					
	0	1	2	3	4	5
	Bénéfices annuels					
1	1	3	4	5	5.5	6
2	0.5	2.5	4	5	6	6.5
3	2	4	5.5	6.5	7	7.5

Le problème de décision comporte trois étapes, une pour chaque usine : x_k représente le montant disponible pour les investissements dans le site k et la décision u_k porte sur le montant à investir pour l'amélioration de cete usine. La fonction de transfert associée à l'usine k est $f_k(x_k, u_k) = x_k - u_k$.
En prenant la centaine de milliers d'euros comme unité monétaire, $S_k = \{0, 1, 2, 3, 4, 5\}$, et l'état initial est $x_1 = 5$. L'ensemble des décisions admissibles dans l'état x_k est contraint à ne pas dépasser le budget disponible. Finalement, notant $g_k(x_k, u_k)$ les bénéfices annuels du site k après un investissement u_k, et $g_4(x_4)$ les bénéfices finalement dégagés et non réinvestis, le problème dynamique déterministe revient à déterminer une suite de décisions admissibles solution de :

$$J^*(5) = \max_{u_1, u_2, u_3} \left(\sum_{k=1}^{3} g_k(x_k, u_k) + g_4(x_4) \right)$$

En supposant, pour simplifier, que les montants non investis ne produisent pas d'intérêt, la résolution suit alors l'algorithme précédent :

1. initialisation : $J_4(x_4) = g_4(x_4) = x_4$
2. propagation :
   – résoudre

$$\begin{aligned} J_3(x_3) &= \max_{u_3 \in U_3(x_3)} \left(g_3(x_3, u_3) + J_4(f_3(x_3, u_3))\right) \\ &= \max_{u_3 \in U_3(x_3)} \left(g_3(x_3, u_3) + x_3 - u_3\right) \end{aligned}$$

Pour $x_3 \in \{0 \cdots 5\}$:

$$\begin{aligned} J_3(0) &= g_3(0, 0) \\ J_3(0) &= 2 \end{aligned}$$

$$\begin{aligned} J_3(1) &= \max_{u_3 \in \{0,1\}} \left(g_3(1, u_3) + (1 - u_3)\right) \\ &= max\left(g_3(1, 0) + 1, g_3(1, 1)\right) \\ &= max\left(2 + 1, 4)\right) \\ J_3(1) &= 4 \end{aligned}$$

$$\begin{aligned} J_3(2) &= \max_{u_3 \in \{0,2\}} \left(g_3(2, u_3) + (2 - u_3)\right) \\ &= max\left(g_3(2, 0) + 2, g_3(2, 1) + 1, g_3(2, 2)\right) \\ &= max\left(2 + 2, 4 + 1, 5.5)\right) \\ J_3(2) &= 5.5 \end{aligned}$$

$$\begin{aligned} J_3(3) &= \max_{u_3 \in \{0,3\}} \left(g_3(3, u_3) + (3 - u_3)\right) \\ &= max\left(g_3(3, 0) + 3, g_3(3, 1) + 2, g_3(3, 2) + 1, g_3(3, 3)\right) \\ &= max\left(2 + 3, 4 + 2, 5.5 + 1, 6.5)\right) \\ J_3(3) &= 6.5 \end{aligned}$$

$$\begin{aligned} J_3(4) &= \max_{u_3 \in \{0,4\}} \left(g_3(4, u_3) + (4 - u_3)\right) \\ &= max(g_3(4, 0) + 4, g_3(4, 1) + 3, g_3(4, 2) + 2, g_3(4, 3) + 1, \\ &\quad g_3(4, 4)) \\ &= max\left(2 + 4, 4 + 3, 5.5 + 2, 6.5 + 1, 7)\right) \\ J_3(4) &= 7.5 \end{aligned}$$

$$
\begin{aligned}
J_3(5) &= \max_{u_3 \in \{0,5\}} \left(g_3(5,u_3) + (5-u_3)\right) \\
&= max(g_3(5,0)+5, g_3(5,1)+4, g_3(5,2)+3, g_3(5,3) \\
&+ 2, g_3(5,4)+1, g_3(5,5)) \\
&= max\left(2+5, 4+4, 5.5+3, 6.5+2, 7+1, 7.5\right)) \\
J_3(5) &= 8.5
\end{aligned}
$$

et la table optimale pour le site 3 est

x_3	0	1	2	3	4	5
$J_3(x_3)$	2	4	5.5	6.5	7.5	8.5
$\mu_3^*(x_3)$	0	1	2	2 ;3	2 ;3	2 ;3

– de même pour le site 2 ($x_2 \in \{0 \cdots 5\}$), on résoud

$$
\begin{aligned}
J_2(x_2) &= \max_{u_2 \in U_2(x_2)} \left(g_2(x_2,u_2) + J_3(f_2(x_2,u_2))\right) \\
&= \max_{u_2 \in U_2(x_2)} \left(g_2(x_2,u_2) + J_3(x_2-u_2)\right)
\end{aligned}
$$

ce qui donne le tableau de décision et de coûts optimaux :

x_2	0	1	2	3	4	5
$J_2(x_2)$	2.5	4.5	6.5	8	9.5	10.5
$\mu_2^*(x_2)$	0	0 ;1	1	1 ;2	2	2 ;3

– enfin, le budget disponible correspondant à $x_1 = 5$, on peut se limiter dans le calcul de $J_1(x_1)$ à cette valeur, et :

$$
\begin{aligned}
J_1(5) &= \max_{u_5 \in \{0,5\}} \left(g_1(5,u_1) + J_2(5-u_1)\right) \\
&= max\left(1+10.5, 3+9.5, 4+8, 5+6.5, 5.5+4.5, 6+2.5\right) \\
J_1(5) &= 12.5
\end{aligned}
$$

et la décision optimale est donc $\mu_1^*(5) = 1$

3. conclusion : ainsi, le bénéfice maximal que l'entreprise peut espérer de ses investissements est de 12.5 centaines de milliers d'euros, et pour cela, elle doit investir cent mille euros dans le site 1 ($\mu_1^*(5) = 1$), et les quatre cent mille euros restant doivent être investis d'abord à hauteur de 200000 dans l'usine 2 ($\mu_2^*(4) = 2$), puis à hauteur de 200000 dans l'usine 1 ($\mu_3^*(2) = 2$).

Remarque :
Ce problème peut être résolu par des techniques de programmation linéaire, en tant que problème d'allocation de ressources. Cependant, pour ce faire, il doit répondre à des contraintes sur les ensembles U_k et le fonctions g_k dont la programmation dynamique s'affranchit.

3.2 Relation avec les problèmes de plus courts chemins

3.2.1 Equivalence

Nous allons illustrer ici par construction l'équivalence qui existe entre les problèmes de plus courts chemins et les problèmes de décisions déterministes à espace d'état fini.
Soit donc tout d'abord un problème dynamique déterministe dans lequel l'espace des états S_k est fini pour toute étape k. Définissons un graphe $G = (E, V)$ de la manière suivante :

- pour chaque étape $k \in \{1 \cdots N\}$, un sommet $e \in E$ est associé à chaque état $x_k \in S_k$
- pour chaque décision $u_k \in U_k(x_k)$, un arc de $v \in V$ est créé, reliant le sommet associé à x_k à celui associé à $x_{k+1} = f_k(x_k, u_k)$. Le poids de v est $g_k(x_k, u_k)$.
- un dernier sommet e_f est ajouté à E, de manière artificielle, pour prendre en compte le coût terminal $g_{N+1}(x_{N+1})$: chaque sommet correspondant à un état x_{N+1} dans lequel le processus peut s'arrêter est relié à e_f par un arc dont le poids est $g_{N+1}(x_{N+1})$.

La recherche d'une politique de décision optimale revient alors à déterminer dans G un plus court chemin (dans un problème de minimisation) entre le sommet associé à x_1 et e_f.

L'implication inverse, qui consiste à montrer que tout problème de plus court chemin peut être modélisé par un programme dynamique déterministe à espace d'état fini, est illustré à l'aide d'un un graphe orienté $G = (E, V)$, $E = \{e_1 \cdots e_n\}$, de matrice des poids associée $C = (c_{i,j})_{1 \leq i,j \leq n}$, $c_{i,j}$ étant le poids de l'arc orienté (e_i, e_j) (avec par convention $c_{i,j} = +\infty$ si e_i et e_j ne sont pas reliés, et $c_{i,i} = 0$). On se propose de trouver, pour chaque sommet $e_i, i \leq i \leq n$ le plus court chemin reliant e_1 à e_i.
Si l'on suppose que G ne contient pas de circuit de longueur négative (sans quoi le problème n'a aucune solution), un plus court chemin contient au plus $n - 1$ arcs, et il est donc possible de formuler le problème en un problème de décisions à $n-1$ étapes. A l'étape k, seuls les chemins utilisant k arcs sont considérés. Pour $i \in \{1 \cdots n\}$, on note $J_k(i)$ la longueur d'un plus court chemin de e_1 à e_i utilisant k arcs, avec par convention $J_k(i) = +\infty$ s'il n'est pas possible de relier e_1 à e_i de cette manière. Enfin, si le plus court chemin de e_1 à e_i contient $p < k$ arcs de G, il est complété par $k - p$ mouvements dégénérés correspondant à des boucles de poids nul, par exemple autour de e_i.
Par le principe d'optimalité, si le plus court chemin de e_1 à e_i en k arcs visite e_j juste avant e_i, le sous-chemin de e_1 à e_j doit correspondre au plus court chemin

entre ces deux sommets en $k-1$ arcs. Ainsi :

$$(\forall k \in \{2 \cdots n-1\}) J_k(i) = \min_{j \in \{2 \cdots n\}} \{J_{k-1}(j) + c_{j,i}\}$$

et

$$(\forall i \in \{2 \cdots n\}) J_1(i) = c_{1,i}$$

et l'on se ramène à un problème de programmation dynamique déterministe dans lequel, pour une étape k, la décision optimale $\mu^*(i)$ correspond au prédécesseur immédiat de e_i dans le plus court chemin de longueur $J_k(i)$. On remarque au passage que les étapes se déroulent dans le sens croissant, ce qui n'est possible que pour les problèmes déterministes.

Exemple :
Soit le graphe G à 5 sommets donné par sa matrice de poids :

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

Alors : $(\forall (i \in \{2 \cdots 5\}) J_1(i) = c_{1,i}$ et la table optimale est :

i	2	3	4	5
$J_1(i)$	1	4	∞	∞
$\mu_1(i)$	1	1	-	-

Puis :

$$J_2(i) = \min_{j \in \{2 \cdots 5\}} \{J_1(j) + c_{j,i}\}$$

i	2	3	4	5
$J_2(i)$	1	4	3	6
$\mu_2(i)$	2	3	2	3

$$J_3(i) = \min_{j \in \{2 \cdots 5\}} \{J_2(j) + c_{j,i}\}$$

i	2	3	4	5
$J_3(i)$	1	2	3	5
$\mu_3(i)$	2	4	4	4

$$J_4(i) = \min_{j \in \{2 \cdots 5\}} \{J_3(j) + c_{j,i}\}$$

i	2	3	4	5
$J_4(i)$	1	2	3	4
$\mu_4(i)$	2	3	4	3

et les plus courts chemins sont donc (e_1, e_2), (e_1, e_2, e_4, e_3), (e_1, e_2, e_4) et $(e_1, e_2, e_4, e_3, e_5)$ de longueurs respectives 1, 2, 3 et 4.

3.2.2 Formalisation : algorithme de Floyd

Le graphe $G = (E, V)$ à n sommets est représenté comme un tableau C d'entiers positifs de taille $n \times n$. $c_{i,j}$ est le coût de l'arc orienté (e_i, e_j), avec par convention $c_{i,j} = +\infty$ si e_i et e_j ne sont pas reliés. Le coût d'un chemin est la somme des coûts des arcs de ce chemin. Considérons un chemin de coût minimal entre e_i et e_j et $e_k \neq e_i, e_j$ un sommet intermédiaire sur ce chemin : les sous-chemins de ce chemin, de e_i à e_k et de e_k à e_j sont aussi de coût minimal (sinon, en remplaçant un de ces sous-chemins par un chemin de moindre coût de mêmes extrémités, on diminuerait le coût du chemin de e_i à e_j, ce qui est impossible). Ce problème satisfait donc au principe d'optimalité.
L'algorithme de Floyd est une méthode ascendante, qui calcule successivement pour k croissant de 1 à n, pour tous les couples de sommets (e_i, e_j), les chemins minimaux de e_i à e_j parmi ceux dont les sommets intermédiaires sont dans $\{1 \cdots k\}$, chemins appelés les k-chemins. Notons d_{ij}^k le coût d'un chemin de e_i à e_j minimal parmi les k-chemins. On doit calculer d_{ij}^n à partir de $d_{ij}^0 = c_{ij}$.
Considérons un k-chemin de coût minimum. S'il ne contient pas e_k, c'est un $(k-1)$-chemin de coût minimum ; s'il contient e_k, on peut le diviser en deux sous-chemins de e_i à e_k et de e_k à e_j, qui sont eux-mêmes des $k-1$-chemins, et par le principe d'optimalité, chacun de ces sous-chemins est un $(k-1)$-chemin de coût minimal ; comme l'un des deux cas se produit, on a la relation :

$$d_{ij}^k = min\left(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right)$$

On peut interpréter cette formule récurrente comme une fonction récursive en i,j,k ; une exécution récursive conduirait à des invocations multiples. L'algorithme de Floyd procède au contraire par une évaluation ascendante.

4 Programmes dynamiques stochastiques

4.1 Définition du problème

Dans ce type de problème, on cherche à minimiser l'espérance des coûts totaux et la politique admissible de décisions optimale correspondante qui, contrairement au cas déterministe, ne fournit pas la suite de décisions mais guide simplement la mise en oeuvre des décisions de manière séquentielle. Pour un système initialement dans l'état x_1, la décision $u_1 = \mu_1^*(x_1)$ est retenue. Après réalisation de la variable aléatoire ω_1, le nouvel état du système x_2 est observé, et la décision

$u_2 = \mu_2^*(x_2)$ peut alors être calculée. L'application itérée de ce processus, jusqu'à la période N, fournit le vecteur π^*.

4.2 Optimisation sur l'ensemble des politiques admissibles

L'optimisation sur l'ensemble des politiques admissibles est bien différente de celle effectuée dans le cadre déterministe, sur l'ensemble des suites de décisions. Pour s'en convaincre, examinons l'exemple suivant.

4.2.1 Position du problème

Un joueur d'échec A doit disputer une rencontre en deux parties et désire maximiser ses chances de gain. Chaque partie rapporte un point au vainqueur et zéro au perdant, et un match nul rapporte 0.5 point à chaque participant. Si chaque joueur remporte une partie, ils jouent une partie de mort subite. Connaissant bien les tactiques de son adversaire, A se propose d'adapter son style de jeu en fonction de ces dernières : il peut jouer par l'attaque, lui donnant une probabilité $p_a > 0$ de gagner et $(1 - p_a) > 0$ de perdre, ou par la défense, lui donnant une probabilité $p_d > 0$ de nul et $(1 - p_d) > 0$ de perdre. Le problème de A est de déterminer une stratégie de sélection de son style de jeu maximisant sa probabilité de gagner la rencontre.

4.2.2 Modélisation du problème

Le joueur ne pouvant gagner avec un jeu uniquement défensif, il doit nécessairement adopter un style d'attaque au maximum à la troisième partie pour espérer remporter la rencontre. Ainsi, le problème se ramène au choix du style de jeu uniquement pour les deux premières parties.
Notons x_k la variable d'état représentant le score avant le début de la partie k, et donc $S_1 = \{0/0\}$. De même x_2 prend ses valeurs dans $S_2 = \{1/0, 0.5/0.5, 0/1\}$, et x_3 dans $S_3 = \{2/0, 1.5/0.5, 1/1, 0.5/1.5, 0/2\}$.
La décision u_k est à valeurs discrètes et vaut a si le style choisi est l'attaque, d si c'est la défense.
La fonction de transfert du système $x_{k+1} = f_k(x_k, u_k, \omega_k)$ dépend, outre de l'état du système et de la décision prise à l'étape k, de la variable aléatoire ω_k, donnant le résultat du match à l'étape k, de loi de probabilité dépendant de u_k. Ainsi par exemple :

$$x_2 = \begin{cases} 0.5/0.5 & \text{avec probabilité } p_d \text{ si } u_1 = d \\ 0/1 & \text{avec probabilité } 1 - p_d \text{ si } u_1 = d \\ 0/1 & \text{avec probabilité } 1 - p_a \text{ si } u_1 = a \\ 1/0 & \text{avec probabilité } p_a \text{ si } u_1 = a \end{cases}$$

La probabilité de gain de A est donnée par le coût final :

$$g_3(x_3) = \begin{cases} 1 & \text{si } x_3 = 2/0 \text{ ou } x_3 = 1.5/0.5 \\ p_a & \text{si } x_3 = 1/1 \\ 0 & \text{si } x_3 = 0/2 \text{ ou } x_3 = 0.5/1.5 \end{cases}$$

La politique admissible optimale $\pi^* = (\mu_1^*, \mu_2^*)$ est donnée par la maximisation

$$J^*(x_1) = \max_{\Pi} \mathbb{E}_{\omega_1,\omega_2} \left(g_3(x_3) \right),$$

4.2.3 Exemple de résolution

Dans le cas où $p_a = 0.45$ et p_d=0.9, l'algorithme se déroule comme suit :

1. initialisation : $(\forall x_3 \in S_3) J_3(x_3) = g_3(x_3)$
2. propagation :
   – résoudre

$$(\forall x_2 \in S_2) J_2(x_2) = \max_{u_2 \in \{a,d\}} \mathbb{E}_{\omega_2} \left(J_3(f_2(x_2, u_2, \omega_2)) \right.$$

On a alors :

$$\begin{aligned} J_2(1/0) &= \max_{u_2 \in \{a,d\}} \mathbb{E}_{\omega_2} \left(J_3(f_2(1/0, u_2, \omega_2)) \right. \\ &= max(p_a J_3(2/0) + (1 - p_a) J_3(1/1), p_d J_3(1.5/0.5) \\ &+ (1 - p_d) J_3(1/1)) \\ &= max(0.45 * 1 + 0.55 * 0.45, 0.9 * 1 + 0.1 * 0.45) \\ J_2(1/0) &= 0.945 \end{aligned}$$

et de même

$$\begin{aligned} J_2(0.5/0.5) &= 0.45 \text{ réalisée pour } u_2 = a \\ J_2(0/1) &= 0.2025 \text{ réalisée pour } u_2 = a \end{aligned}$$

La politique de décisions optimales est donc :

$$\mu_2^*(1/0) = d, \mu_2^*(0.5/0.5) = a, \mu_2^*(0/1) = a$$

– résoudre

$$(\forall x_1 \in S_1) J_1(x_1) = \max_{u_1 \in \{a,d\}} \mathbb{E}_{\omega_1} \left(J_2(f_1(x_1, u_1, \omega_1)) \right.$$

avec $x_1 = (0/0)$, il vient :

$$\begin{aligned} J_1(0/0) &= \max_{u_1 \in \{a,d\}} \mathbb{E}_{\omega_1} \left(J_2(f_2(0/0, u_1, \omega_1)) \right) \\ &= max(p_a J_2(1/0) + (1 - p_a) J_3(0/1), p_d J_2(0.5/0.5) + \\ &(1 - p_d) J_2(0/1)) \\ &= max(0.45 * 0.945 + 0.55 * 0.2025, 0.9 * 0.45 + \\ &0.1 * 0.2025) \\ J_2(1/0) &= 0.536625 \end{aligned}$$

3. conclusion : ainsi, la stratégie optimale pour A consiste à adopter un style agressif pour les deux parties, sauf s'il mène au score à la fin de la première partie. En appliquant une telle politique, et avec les valeurs de p_a et p_d définies dans l'exemple, sa probabilité de gagner la rencontre est d'environ 0.537.

5 Généralisation des fonctions de coût

Jusqu'alors, les fonctions de coût présentées s'obtenaient en sommant les coûts d'étapes. Outre ces fonctions additives, il existe deux autres grandes classes de fonctions pour lesquelles la programmation dynamique peut être appliquée.

5.1 Fonctions multiplicatives

Si les coûts $g_k(x_k, u_k, \omega_k)$ ne prennent que des valeurs positives ou nulles pour toute étape k, et si les valeurs du coût final $g_{N+1}(x_{N+1})$ sont également positives ou nulles pour tout état x_{N+1}, il est possible d'optimiser une espérance des coûts totaux de la forme :

$$\mathbb{E}_{\omega_1 \cdots \omega_N} \left(g_{N+1}(x_{N+1}) \times \prod_{k=1}^{N} g_k(x_k, u_k, \omega_k) \right)$$

L'algorithme de programmation dynamique s'applique alors directement, en changeant la relation de récurrence exprimant les $J_k(x_k)$ en :

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}_{\omega_k} \left(g_k(x_k, u_k, \omega_k) \times J_{k+1}(f_k(x_k, u_k, \omega_k)) \right)$$

Dans le cas déterministe, il est facile de démontrer (en passant au log) que les modèles additifs et multiplicatifs sont équivalents.

5.2 Fonctions minimax ou maximin

Dans le cas des fonctions minimax, la performance globale est égale à

$$\mathbb{E}_{\omega_1 \cdots \omega_N} \left(max \left(g_k(x_k, u_k, \omega_k)_{k \in \{1 \cdots N+1\}}, g_{N+1}(x_{N+1}) \right) \right)$$

Là encore, l'algorithme de programmation dynamique s'applique, simplement en changeant la relation de récurrence en :

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}_{\omega_k} \left(max \left(g_k(x_k, u_k, \omega_k), J_{k+1}(f_k(x_k, u_k, \omega_k)) \right) \right)$$

Dans le cas des fonctions maximin, il suffit d'inverser l'ordre du min et du max.

5.3 Inclusion d'un facteur d'actualisation

En plus de ces trois grandes catégories de fonctions, il est possible pour chacune d'entre elles d'inclure un **facteur d'actualisation** (ou **facteur d'escompte**) $\alpha \in]0, 1]$ dans l'algorithme de programmation dynamique. L'objectif est alors de déterminer une politique de décisions optimisant la somme, le produit, le minimax ou le maximin des coûts escomptés. Pour une fonction de coût additive par exemple, l'optimisation portera sur :

$$\mathbb{E}_{\omega_1 \cdots \omega_N} \left(\alpha^N g_{N+1}(x_{N+1}) + \sum_{k=1}^{N} \alpha^{k-1} g_k(x_k, u_k, \omega_k) \right)$$

La relation de récurrence de l'algorithme devient alors dans ce cas :

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}_{\omega_k} \left(g_k(x_k, u_k, \omega_k) + \alpha J_{k+1}(f_k(x_k, u_k, \omega_k)) \right)$$

Cette relation peut être vue comme celle associée à un programme dynamique stochastique, différant du problème initial par la présence, en fin de chaque période, d'un choix aléatoire provoquant l'arrêt du processus avec probabilité $1 - \alpha$. Un processus arrêté n'entraînant plus aucun coût, l'espérance de la somme des coûts pour le sous-problème débutant dans l'état x_k à l'année k, et s'arrêtant à la fin de celle-ci avec la probabilité $1 - \alpha$ est exactement la relation de récurrence précédente.

Exemple : remplacement d'une machine
Une entreprise a besoin d'utiliser une machine pour les N=5 prochaines années. Pour l'instant, elle dispose d'une machine neuve. En fin d'année, le chef d'entreprise décide s'il veut garder sa machine ou la revendre et la remplacer par une

neuve. La durée de vie d'une machine est de 3 ans maximum et le pris d'achat p est de 50000 euros. Les revenus annuels r_i, les coûts de maintenance c_i et le prix de revente p_i sont (en milliers de francs) :

i	Age i en début d'année		
	0	1	2
$r(i)$	40	30	15
$c(i)$	5	7	11
$p(i)$	25	12	5

Enfin, la machine finale est revendue à la fin des 5 ans.
Le chef d'entreprise souhaite adopter la stratégie de remplacement qui lui fasse maximiser son profit pour les cinq prochaines années.
Notons x_k l'âge de la machine en début d'année k, u_k la décision qui consiste à garder (g) ou revendre (r) la machine en fin d'année k. La fonction de transfert du système est alors donnée par :

$$x_{k+1} = \begin{cases} x_k + 1 & \text{si } u_k = g \\ 0 & \text{si } u_k = r \end{cases}$$

Le profit pendant l'année k est égal à
– $g_k(x_k, u_k) = r(x_k) - c(x_k)$ si la machine est gardée pour l'année suivante
– $g_k(x_k, u_k) = r(x_k) + p(x_k) - c(x_k) - p$ sinon
La nécessité d'introduire une actualisation des coûts et des profits provient des intervalles de temps relativement grands (l'année) entre chaque décision. Ainsi, le profit total de l'entreprise, pour la suite de décisions $(u_k)_{1 \leq k \leq 5}$ est

$$\sum_{k=1}^{5} \alpha^{k-1} g_k(x_k, u_k)$$

Considérant qu'aucune variable aléatoire de bruit ω_k ne vient perturber le système, et que $\alpha = 0.9$, l'algorithme se déroule alors comme suit :

1. initialisation : en fin de cycle, l'énoncé impose $u_5 = r$ et aucune machine ne doit être rachetée. Donc $J_5(x_5) = r(x_5) + p(x_5) - c(x_5)$
2. propagation :
   – résoudre

$$J_4(x_4) = \max_{u_4 \in U_4(x_4)} \left(g_4(x_4, u_4) + \alpha J_5(f_4(x_4, u_4)) \right)$$

et ainsi

$$\begin{aligned} J_4(0) &= \max_{u_4 \in \{g,r\}} \left(g_4(0,u_4) + \alpha J_5(f_4(0,u_4))\right) \\ &= max\left(g_4(0,g) + \alpha J_5(1), g_4(0,r) + \alpha J_5(0)\right) \\ &= max\left(40 - 5 + 0.9(30 + 12 - 7), 40 + 25 - 5 - 50 + 0.9(40 + 25 - 5)\right) \\ J_4(0) &= 66.5 \end{aligned}$$

et de même

$$\begin{aligned} J_4(1) &= 39 \\ J_4(2) &= 13 \end{aligned}$$

et donc la politique optimale pour la quatrième année consiste à remplacer la machine si elle a un ou deux ans, et la garder si elle est neuve.

– un raisonnement identique sur les troisième et seconde année aboutissent à la politique de décision suivante : la machine est conservée la troisième année si elle est neuve ($J_3(0) = 70.1$), remplacée sinon ($J_3(1) = 44.85$, $J_3(2) = 18.85$), et idem pour la seconde année ($J_2(0) = 75.365$, $J_2(1) = 48.09$, $J_2(2) = 22.09$)

– résoudre

$$J_1(x_1) = \max_{u_1 \in U_1(x_1)} \left(g_1(x_1,u_1) + \alpha J_2(f_1(x_1,u_1))\right)$$

avec $x_1 = 0$, seul état initial permis, on a

$$\begin{aligned} J_1(0) &= \max_{u_1 \in \{g,r\}} \left(g_1(0,u_1) + \alpha J_2(f_1(0,u_1))\right) \\ &= max\left(g_1(0,g) + \alpha J_2(1), g_1(0,r) + \alpha J_2(0)\right) \\ &= max\left(35 + 0.9 * 48.09, 10 + 0.9 * 75, 8285\right) \\ J_1(0) &= 78,281 \end{aligned}$$

et la politique optimale consiste bien sûr à garder la machine.

3. conclusion : finalement, le chef d'entreprise a intérêt à renouveler sa machine tous les deux ans.

6 Nature des variables

Jusqu'à présent, seuls étaient considérés des ensembles d'état S_k et de décision C_k discrets et mêmes finis. Les méthodes de programmation dynamique restent applicables pour des espaces de définition plus généraux (espaces discrets dénombrables, espaces multidimensionnels, espaces continus), mais posent des difficultés de résolution numérique. En particulier, on est généralement amenés à discrétiser les espaces continus pour accélérer l'algorithme, au prix d'une approximation de la solution optimale.

6.1 Premier exemple : utilisation de variables continues

On discrétise le temps en N intervalles, et on cherche à optimiser la trajectoire d'un missile pour qu'il atteigne une cible à une vitesse V donnée. La vitesse du missile peut être modifiée de Δ_v au début de chaque intervalle de temps, et pour éviter de trop brusques changements de vitesse, Δ_v est pénalisé par un coût de Δ_v^2. A la fin du processus, le missile atteint une vitesse v_f et une pénalité égale à $4(V - v_f)^2$ est appliquée. Le programme dynamique se propose de déterminer la suite de variations de vitesses à appliquer au début de chaque période de temps pour minimiser la somme des pénalités.
On note alors x_k la vitesse du missile au début de l'intervalle de temps k, u_k la vitesse choisie pour cet intervalle. La fonction de transfert est donc simplement $x_{k+1} = u_k$. Le coût associé à une suite de décisions $(u_1 \cdots u_N)$ est alors :

$$\sum_{k=1}^{N} (x_k - u_k)^2 + 4(V - x_{N+1})^2$$

Fixons par exemple $N = 2$. L'algorithme se déroule alors comme suit :

1. initialisation : $(\forall x_3 \in \mathbb{R}^+) J_3(x_3) = 4(V - x_3)^2$
2. propagation :
   – résoudre
   $$J_2(x_2) = \min_{u_2 \in \mathbb{R}} \left((x_2 - u_2)^2 + J_3(u_2) \right)$$
   soit :
   $$J_2(x_2) = \min_{u_2 \in \mathbb{R}} \left((x_2 - u_2)^2 + 4(V - u_2)^2 \right)$$
   et en dérivant la fonction quadratique, on trouve la vitesse optimale :
   $$u_2^* = \frac{4V + x_2}{5}$$

La pénalité minimale pour les intervalles 2 et 3 est alors :

$$J_2(x_2) = \frac{4}{5}(V - x_2)^2$$

– résoudre

$$J_1(x_1) = \min_{u_1 \in \mathbb{R}} \left((x_1 - u_1)^2 + J_2(u_1)\right)$$

soit

$$J_1(x_1) = \min_{u_1 \in \mathbb{R}} \left((x_1 - u_1)^2 + \frac{4}{5}(V - x_2)^2\right)$$

et par annulation de la dérivée

$$u_1^* = \frac{4V + 5x_1}{9}$$

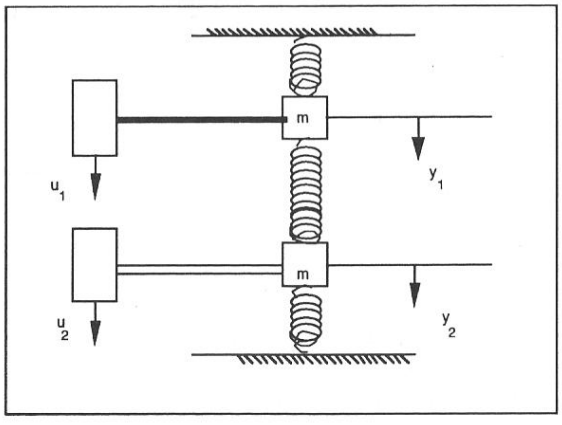
La pénalité minimale pour les intervalles 2 et 3 est alors :

$$J_1(x_1) = \frac{4}{9}(V - x_1)^2$$

3. conclusion : partant d'une vitesse initiale nulle ($x_1 = 0$), et par exemple pour V=900 km/h, les décisions optimales sont alors u_1^*=400 km/h et u_2^*=800 km/h.

6.2 Second exemple : contrôle optimal

Soit le système masses/ressorts suivant :



y_1 et y_2 sont les positions des masses par rapport à la position d'équilibre.
Un moteur (par exemple pour une fusée) est attaché aux deux masses, permettant d'exercer par télécommande une force $u(t)$ dans un sens ou dans l'autre et selon

une intensité variable en fonction du temps.
Les équations du mouvement s'écrivent :

$$\frac{d^2y_1}{dt^2} = -\frac{2k}{m}y_1 + \frac{k}{m}y_2 + u_1$$

$$\frac{d^2y_2}{dt^2} = -\frac{k}{m}y_1 + \frac{2k}{m}y_2 + u_2$$

soit, si $x = (y_1, \frac{dy_1}{dt}, y_2, \frac{dy_2}{dt})$ est le vecteur d'état du système,

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{2k}{m} & 0 & \frac{k}{m} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{m} & 0 & \frac{k}{m} & 0 \end{pmatrix} x + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

soit

$$\frac{dx}{dt} = A(t)x(t) + B(t)u(t)$$

qui est l'équation d'état d'un système linéaire dynamique contrôlé non dépendant du temps. Les matrices $A(t)$ et $B(t)$ ne dépendent pas du temps ici (cas particulier).
Le problème de contrôle optimal est de trouver $u(t)$ minimisant une fonctionnelle, par exemple

$$J(u) = \int_0^T \left(y_1^2 + y_2^2 + \alpha(u_1^2 + u_2^2)\right) dt$$

qui permet de trouver le meilleur compromis dans $[0, T]$ sur l'écart de y_1 et y_2 par rapport à la position d'équilibre et d'un coût énergétique dépensé, dépendant de α.
Plus généralement, le critère fonctionnel quadratique s'écrit :

$$J(u) = \int_0^T \begin{pmatrix} y_1 & \frac{dy_1}{dt} & y_2 & \frac{dy_2}{dt} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 & \frac{dy_1}{dt} & y_2 & \frac{dy_2}{dt} \end{pmatrix} + \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix} \begin{pmatrix} u_1 & u_2 \end{pmatrix} dt$$

soit

$$J(u) = \int_0^T \left((x - x_d)Q(x - x_d) + uR(t)u\right) dt$$

où x_d est une trajectoire de consigne.
En supposant que $[0, T]$ soit subdivisé en N intervalles Δt,

$$\frac{dx}{dt} \approx \frac{x((k+1)\Delta t) - x(k\Delta t)}{\Delta t} = A(t)x(k\Delta t) + B(t)u(k\Delta t)$$

soit si $x(k\Delta t) = x(k)$ et $u(k\Delta t) = u(k)$:

$$x(k+1) = (1 + A(t)\Delta t)x(k) + B(t)\Delta t u(k)$$

et

$$J(u) = \Delta t \sum_{k=0}^{N-1} (x(k) - x_d(k))Q(k)(x(k) - x_d(k)) + u(k)R(k)u(k)$$

que l'on cherche à minimiser.