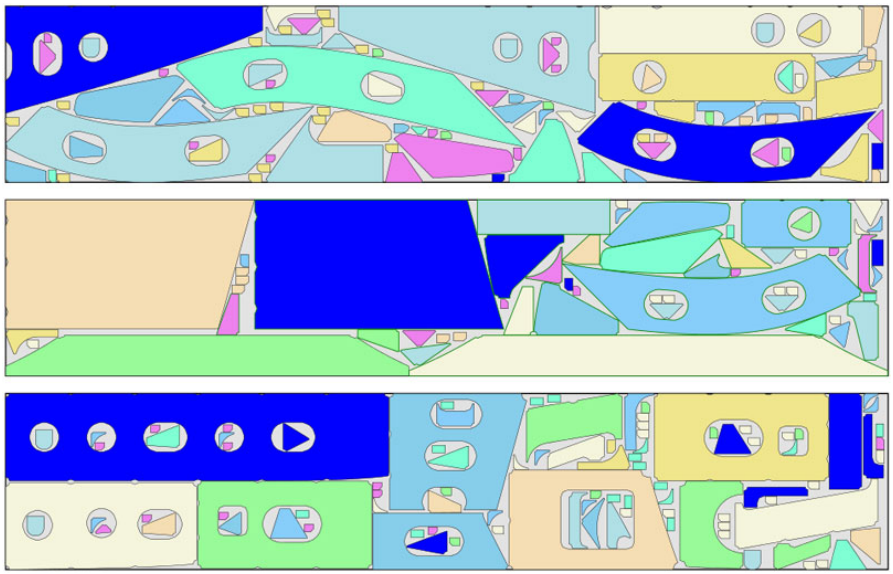
Optimisation Statique

6 septembre 2021

Motivations



Problème de découpage (1, 2, 3D) : Une compagnie produit de grands rouleaux de papier de 10 m de long qui sont ensuite découpés suivant la commande du client. On cherche à minimiser les chutes.

$$\min \sum_{j=1}^{n} x_j$$

$$\text{sous la contrainte } \sum_{j=1}^{n} A_{ij} x_j \geq b_i, \;\; x_j \geq 0, \;\; x_j \in \mathbb{N}.$$

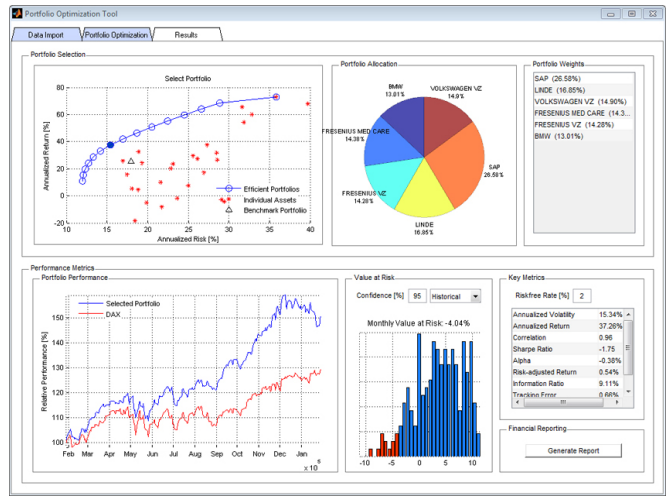
Motivations

Optimisation de portefeuilles : Chaque investisseur sait qu'il y a une relation entre risque et récompense. Pour obtenir de plus grands retours sur investissements, on doit être disposé à prendre de plus grands risques. En fonction de la tolérance d'un investisseur au risque, on peut chercher à optimiser ces investissements suivant :

$$\max r^T w - K\ w^T Q w$$

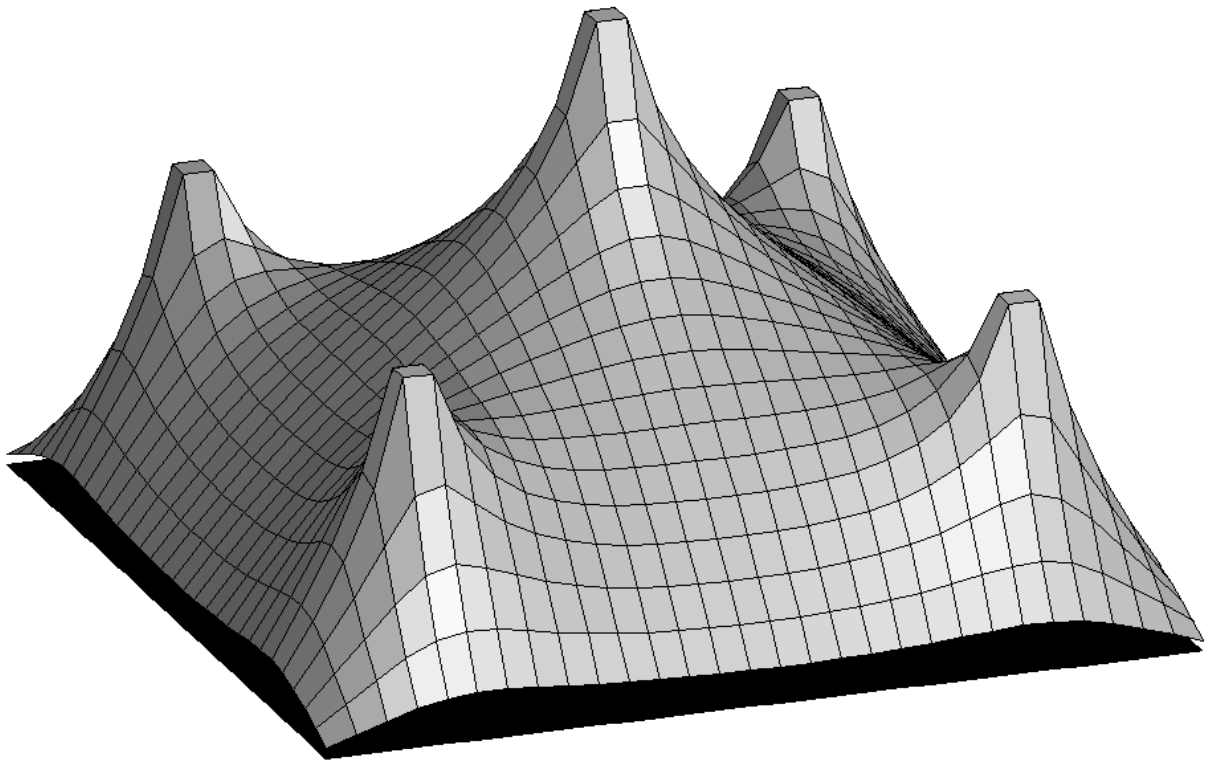
$$\text{sous la contrainte } \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0$$

w_i : Proportion d'action i dans le portefeuille
r : Retour sur investissement
Q : matrice de covariances (Variabilité - Dispersion)
K : facteur d'aversion au risque (assure une pondération entre retour sur inv. et risque accepté)



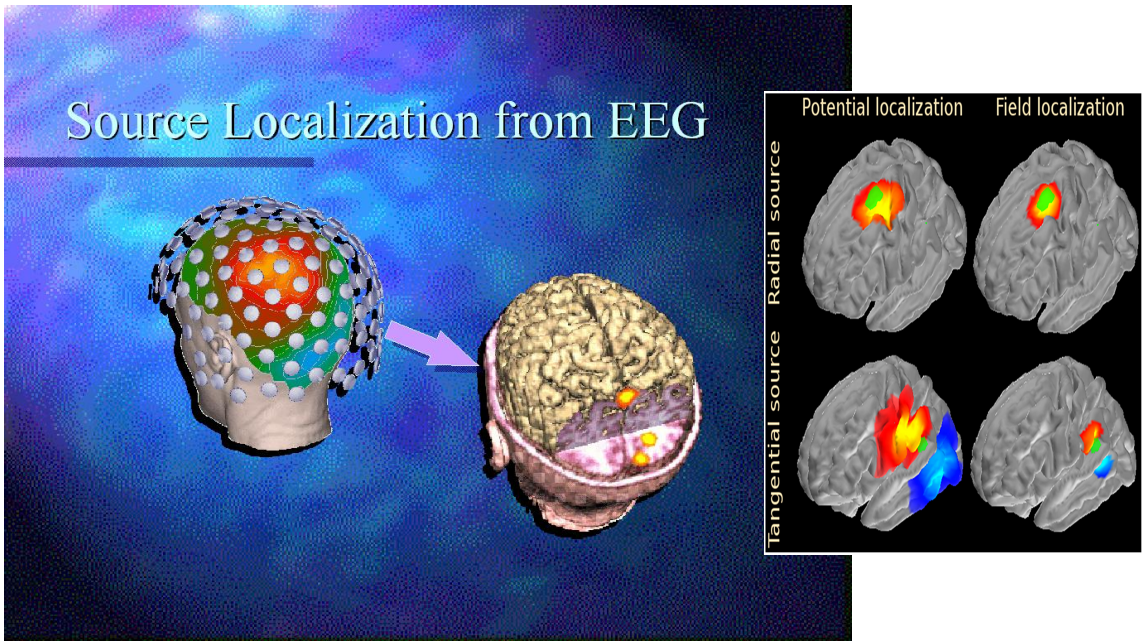
Motivations

Minimisation de forme, surface, volume, ...



Motivations

Localisation de source cérébrale (ou de téléphones mobiles)



Motivations

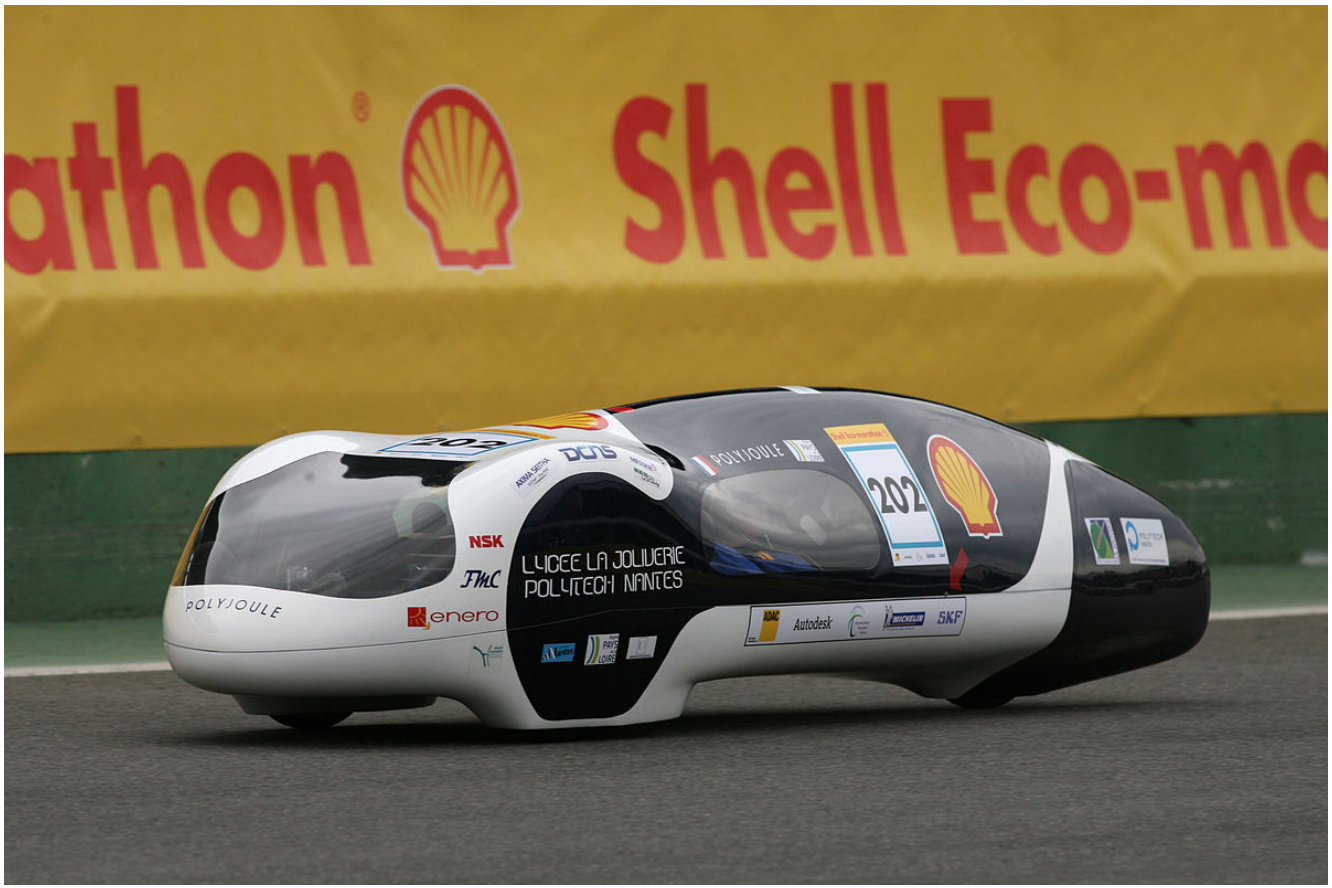
Optimisation d'asservissements : Etude de la robustesse en stabilité ou en performance d'un satellite

Objectif : rendre insensible la visée du spectromètre aux mouvement des panneaux solaires

Motivations

Optimisation de la consommation d'énergie

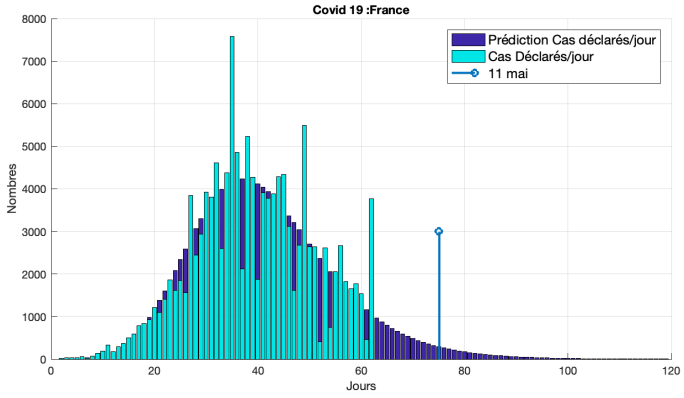
Objectif : Parcourir un 1 km en moins de 3 mn et en minimisant la consommation d'énergie

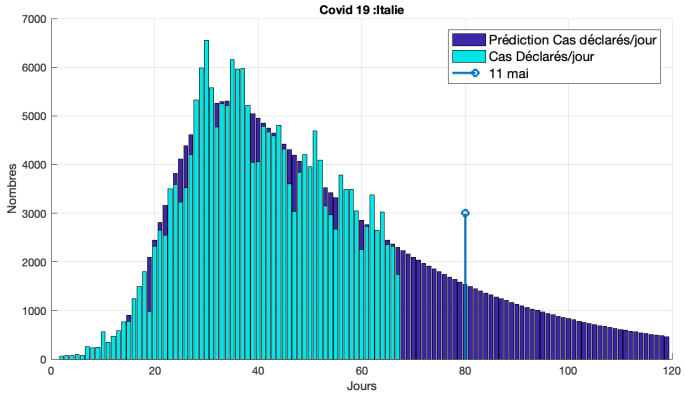


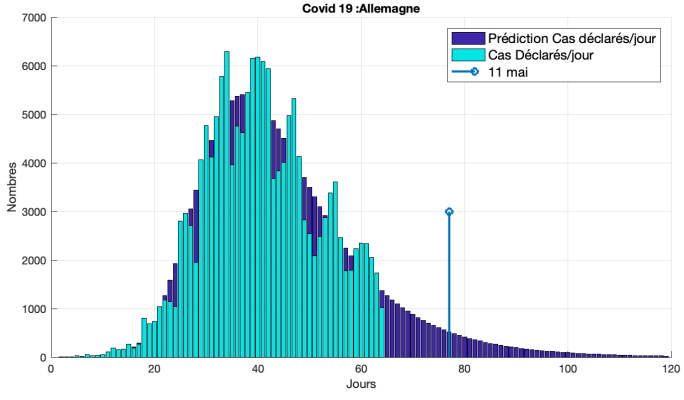
Motivations

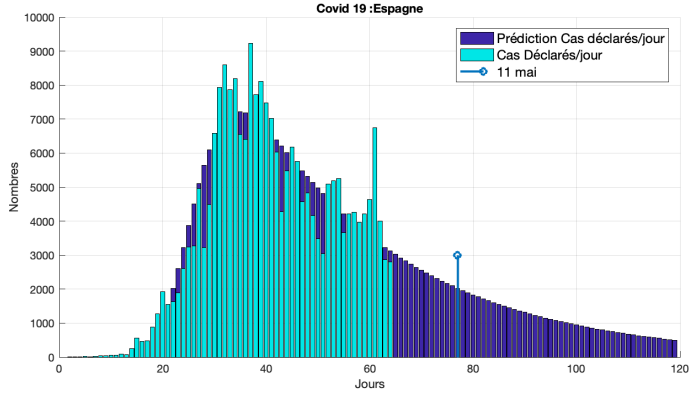
Prévoir l'évolution d'une pandémie

Objectif : Identifier les paramètres d'un modèle dynamique de pandémie à partir de données disponibles









Motivations

Mais encore ...

- Calcul de structure
- Recherche opérationnelle, Logistique : allocation de ressources, gestion de stocks, ordonnancement de taches, plans d'investissements
- Amélioration de procédés : réduction des coûts (énergie, matière), augmentation de la qualité
- Modélisation et Identification paramétrique des systèmes
- Contrôle-commande des processus
- Analyse d'Images, de données
- Apprentissage, Deep Learning.

Motivations

Une même formule :

Trouver le minimum de la fonction scalaire f de la variable $x \in \mathbb{R}^n$

$$\min_{x \in \mathcal{X}} f(x)$$

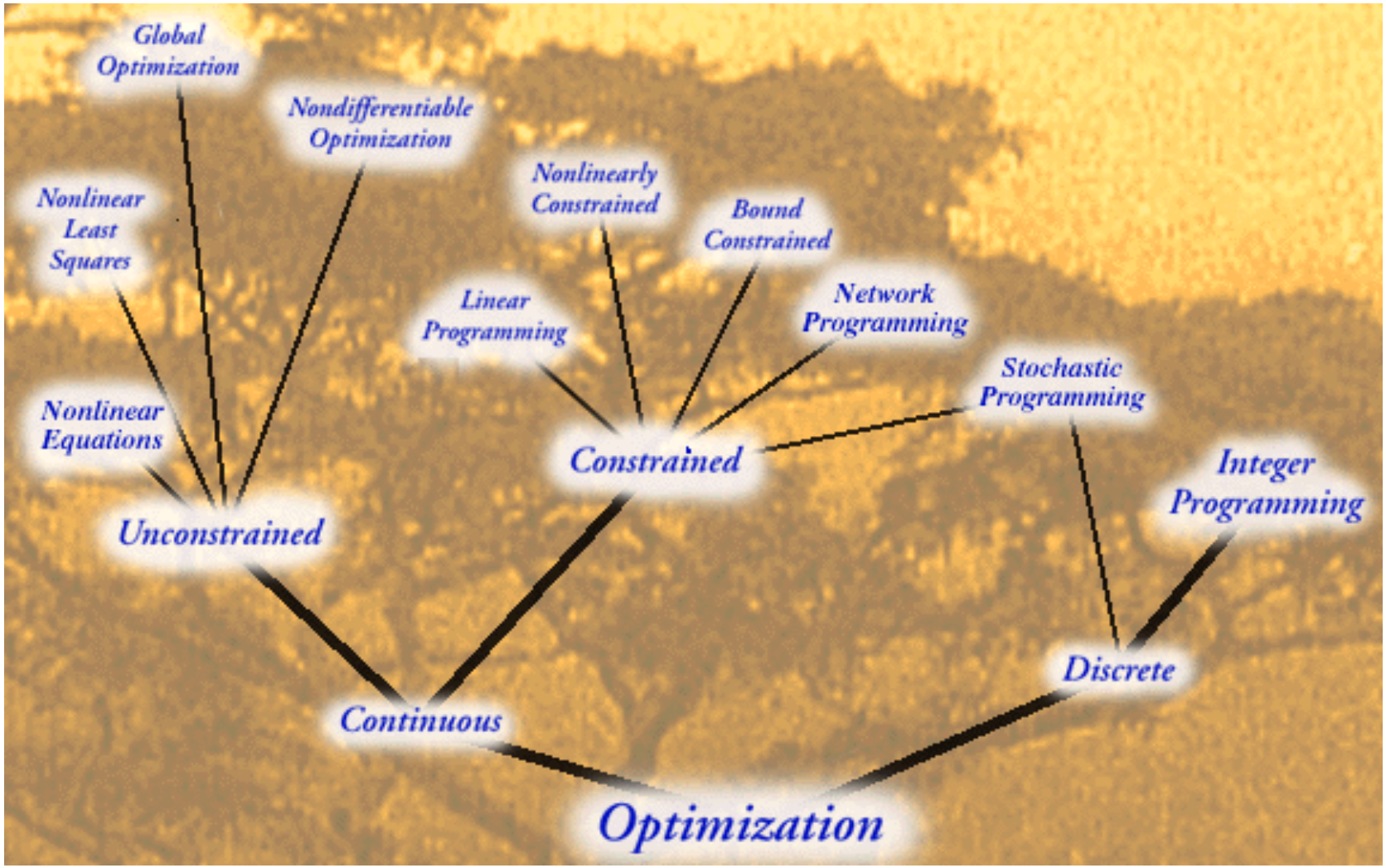
sur un domaine $\mathcal{X} \subset \mathbb{R}^n$

Motivations

Des méthodes qui vont différer suivant :

- la dérivabilité ou non de la fonction f
- la possibilité d'un calcul effectif ou non des dérivées
- le domaine sur lequel est recherché la solution qui peut être continu ou discret
- le type de contraintes (égalités, inégalités)
- la classe du problème : linéaire, quadratique, convexe, non linéaire
- la dimension du problème (nombres d'inconnues)

Motivations



Objectif, Planning, Evaluation

Objectifs :

- Formulation et traduction d'un problème d'optimisation
- Savoir opérer dans le choix d'une méthode numérique de résolution en fonction de la nature du problème
- Mise en œuvre

Planning

- Optimisation Statique
  - 8 h cours (contraintes et sans contraintes)
  - 4 h TD (A préparer)
  - 8 h en plateforme (Programmation Matlab)

Evaluation :

- Compte-rendus Plateformes
- Un examen

Rappels - Notations - Définitions

On considère une fonction f de $\mathbb{R}^n \to \mathbb{R}$
f est de classe C^1 voire C^2 (calcul du Hessien)

- On note :

$$g(x) = \nabla f(x) = \left[\frac{\partial f}{\partial x_1}(x) \ \frac{\partial f}{\partial x_2}(x) \cdots \frac{\partial f}{\partial x_n}(x) \right]^T$$

le gradient de f au point x et

$$H(x) = \nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(x) \\ \vdots & \ddots & & \vdots \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \cdots & & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix}$$

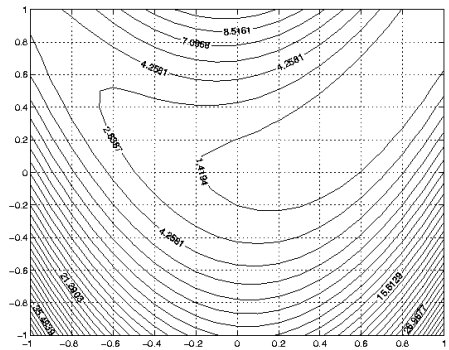
le Hessien de f évalué au point x.

Notations - Définitions - Rappels :

- **Développement de Taylor à l'ordre 2 (D.L.) :**

$$f(x+h) = f(x) + h^T g(x) + \frac{1}{2} h^T H(x) h + o(h^T h)$$

- **Ligne de niveau**
  $L_z = \{x \in \mathbb{R}^n : \ f(x) = z\}$: ensemble des x tels que $f(x) = z$.



- Par définition, **la dérivée de f dans la direction d** est égale à :

$$f'(x;d) = \lim_{\substack{t \to 0 \\ t \in \mathbb{R}}} \frac{f(x+td) - f(x)}{t}$$

- Lorsque g existe (ce qui sera le cas), on a :

$$f'(x;d) = d^T g(x)$$

Rappels

Notation : $g_k = g(x_k)$ $g^* = g(x^*)$
Calcul différentiel matriciel :

$$x \in \mathbb{R}^n,\ y \in \mathbb{R}^n$$
$$\frac{\partial y^T x}{\partial x} = y$$
$$\frac{\partial x^T M x}{\partial x} = (M + M^T)x \quad (= 2Mx \text{ si } M \text{ symétrique})$$

Factorisation QR : Toute matrice $A_{m\times n}$ $m \geq n$ admet une factorisation $A = QR$ avec $Q_{m\times m}$ orthogonale $Q^T = Q^{-1}$ et $R_{m\times n}$ triangulaire supérieure.

Matrice définie positive $M > 0$:

$$\forall x \neq 0,\ x^T M x > 0$$

Si M est symétrique, les valeurs propres sont toutes strictement positives.

Proposition

Si d appartient à la tangente à la courbe de niveau alors

$$f'(x;d) = d^T g(x) = 0$$

$g(x)$ est donc orthogonal à la courbe de niveau, $g(x) \perp L_{f(x)}$.

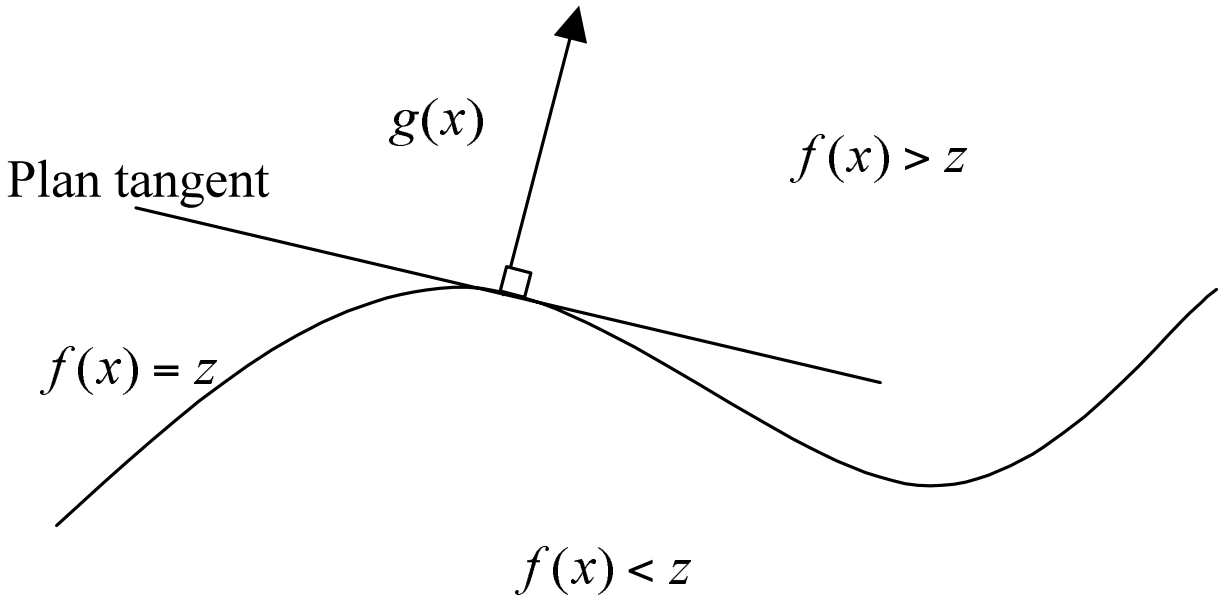


FIGURE: Ligne de niveau et gradient

Démonstration.

- Soit $x_k \to x$ une suite de points situés sur la ligne de niveau L_z.
- $x_k = x + \alpha_k d_k$ avec $\alpha_k = \|x_k - x\|$ et $d_k = \frac{x_k - x}{\|x_k - x\|}$; $\|d_k\| = 1$
  Alors

$$\alpha_k \to 0, \text{ avec } k \to +\infty$$
$$d_k \to d$$

- Montrons que $\lim_{k\to\infty} \frac{f(x_k)-f(x)}{\alpha_k} = f'(x;d)$:

$$\frac{f(x_k) - f(x)}{\alpha_k} = \frac{f(x + \alpha_k d_k) - f(x)}{\alpha_k}$$
$$= g(x)^T d_k + o(\alpha_k)/\alpha_k \quad \text{(D.L.)}$$

- Par passage à la limite, $\lim_{k\to\infty} \frac{f(x_k)-f(x)}{\alpha_k} = f'(x;d)$
- Par ailleurs, comme la suite des points x_k appartient à L_z, $f(x_k) - f(x) = 0, \forall k$ d'où le résultat :

$$0 = \lim_{k\to\infty} \frac{f(x_k) - f(x)}{\alpha_k} = f'(x;d)$$

Problème sans contrainte

Trouver le minimum de la fonction f,

$$\min_{x \in \mathbb{R}^n} f(x)$$

Théorème (CN)

Si x^* est un minimum relatif alors : $g(x^*) = 0$

Démonstration.

D'après le D. L. de f,

$$f(x^* + h) = f(x^*) + h^T g(x^*) + o(\|h\|)$$

Si x^* est un minimum de f, $\forall y \in \mathbb{R}^n$, $\forall t \in \mathbb{R}$ suffisamment petit, on a :

$$f(x^* + ty) - f(x^*) \geq 0$$

Soit, pour $t > 0$,

$$\lim_{t \to 0} \frac{f(x^* + ty) - f(x^*)}{t} \geq 0$$

et pour $t < 0$,

$$\lim_{t \to 0} \frac{f(x^* + ty) - f(x^*)}{t} \leq 0$$

$\Rightarrow \forall y,\ f'(x^*; y) = y^T g(x^*) = 0 \Rightarrow g(x^*) = 0$

Remarque

Attention : le théorème est faux si contrainte sur x.

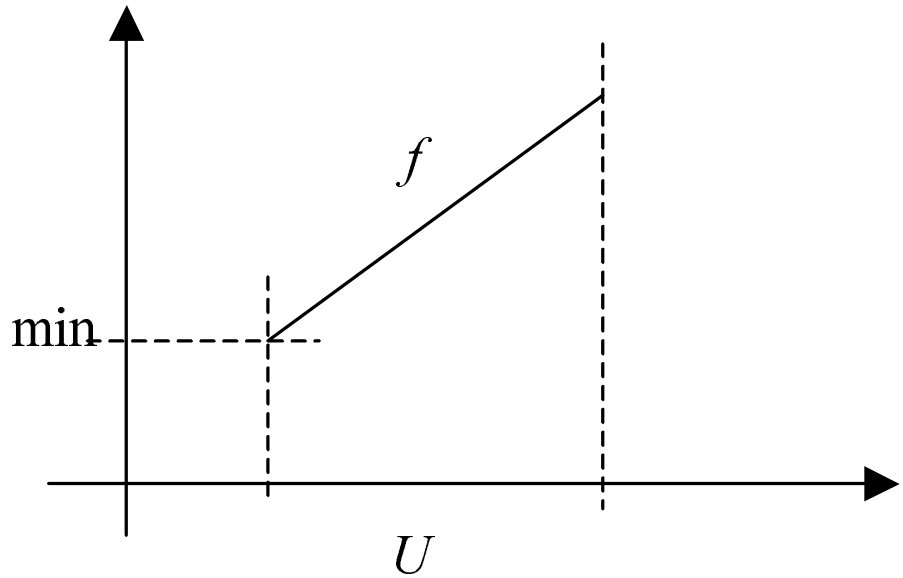


FIGURE: Cas d'une fonction affine sur un intervalle

Remarque

La réciproque est fausse : $g(x) = 0 \not\Rightarrow$ *f atteint un extremum en x*

Théorème (CS)

Si la CN est satisfaite et si $H(x^*) > 0$ alors x^* est un minimum relatif

Démonstration.

Il suffit de développer à l'ordre 2 la fonction f :

$$f(x^* + ty) = f(x^*) + ty^T g(x^*) + \frac{t^2}{2} y^T H(x^*) y + o(t^2)$$

La condition nécessaire de minimum implique

$$f(x^* + ty) - f(x^*) = \frac{t^2}{2} y^T H(x^*) y + o(t^2)$$

Pour t suffisamment petit, comme $y^T H(x^*) y > 0$,

$$f(x^* + ty) - f(x^*) = \frac{t^2}{2} y^T H(x^*) y + o(t^2) > 0$$

i.e. $f(x^* + ty) > f(x^*)$, $\forall y \in \mathbb{R}^n$, $\forall t \in \mathbb{R}$ suffisamment petit. Donc x^* est un minimum. □

Principe des méthodes de descente

Exemple : l'algorithme du gradient

Idée : Résoudre une suite de problèmes d'optimisation mono-dimensionnels

Algorithme général :
Pour $k = 0, 1, 2, \ldots$ et x_0 donné

1. Déterminer une direction de descente d_k depuis la position courante x_k
2. Déterminer une longueur de pas α_k qui "minimise" $f(x_k + \alpha d_k)$ par rapport à α
3. Poser $x_{k+1} = x_k + \alpha_k d_k$,
4. Retourner en 1 tant que le test d'arrêt n'est pas validé.

Fin Pour

Test d'arrêt : On choisit un $\varepsilon > 0$ et l'algorithme s'arrête lorsque
$\|x_{k+1} - x_k\| < \varepsilon$ ou $\|f(x_{k+1}) - f(x_k)\| < \varepsilon$

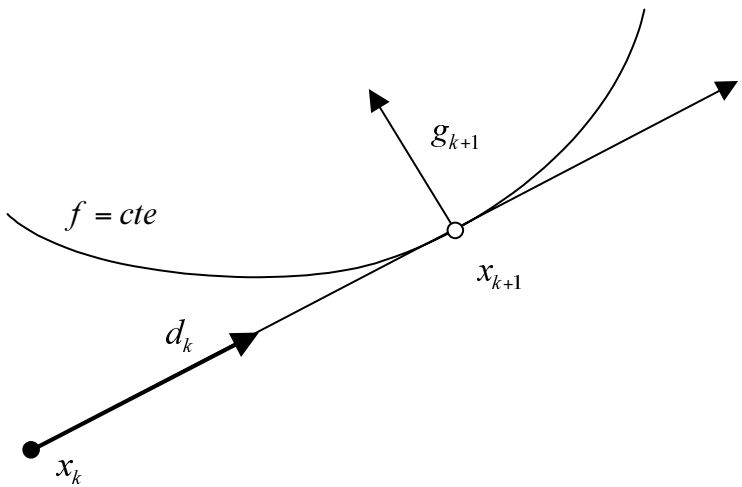
Méthode de descente

- La méthode est dite méthode de descente si la propriété :

$$\left.\frac{\partial f(x_k + \alpha d_k)}{\partial \alpha}\right|_{\alpha=0} = d_k^T g_k = f'(x_k; d_k) < 0$$

  est vérifiée à chaque itération où $g_k = g(x_k)$.
- Si α_k est un minimum de f dans la direction d_k alors la CN implique :

$$f'(x_k + \alpha_k d_k) = d_k^T g_{k+1} = 0$$



Géometriquement : on vient "tangenter" une ligne de niveau

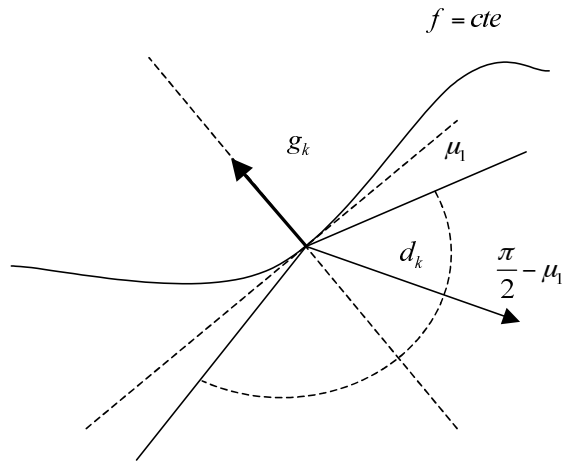
Critère de convergence

Theorem (Critère de convergence)

Si la méthode de descente respecte à chaque itération, les points :

i. *Direction admissible (direction de descente) :*

l'angle $(\widehat{d_k, -g_k}) \leq \frac{\pi}{2} - \mu_1$,
avec $\mu_1 > 0$



ii. Décroissance suffisante de f (Armijo) :
pour $0 < \mu_2 < 1$,

$$f(x_{k+1}) - f(x_k) \leq \mu_2 f'(x_k; \alpha_k d_k) = \mu_2 \alpha_k d_k^T g_k < 0$$

iii. Réduction suffisante de la dérivée f' (Wolfe) : Pour $\mu_2 < \mu_3 < 1$,

$$\left| f'(x_{k+1}; d_k) \right| \leq \mu_3 \left| f'(x_k; d_k) \right|$$

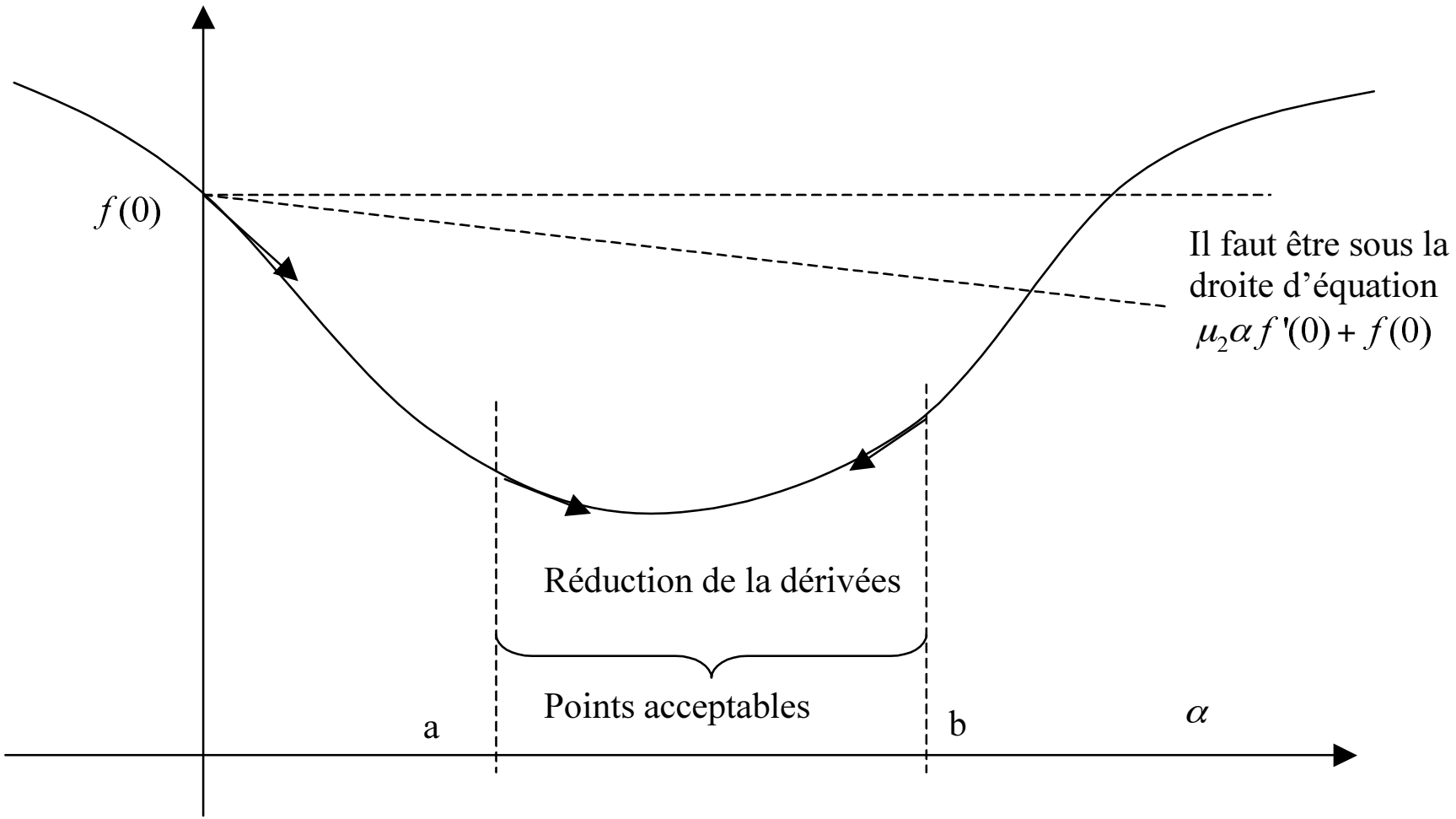
et si g existe et est uniformément continue sur l'ensemble $\{x \in \mathbb{R}^n : f(x) < f(x_0)\}$ alors

ou bien $f_k = f(x_k) \to -\infty$,

ou bien $g_k = g(x_k) \to 0$.

Dans la pratique, on peut choisir par défaut $\mu_2 = 0.001$ et $\mu_3 = 0.9$

Critère de convergence



Algorithme de descente

Comment trouver α ?

A partir d'une direction de descente d_k et d'un point x_k, on procède en 2 phases :

- **Phase 1 :** Détermine un intervalle $I_i = [a_i \ b_i]$ dont on sait qu'il contient un ensemble de points acceptables
- **Phase 2 :** Réduire cet intervalle initial I_i en générant une suite d'intervalles $I_{j+1} \subset I_j$ plus petits et terminer lorsque l'on a trouvé un point acceptable.

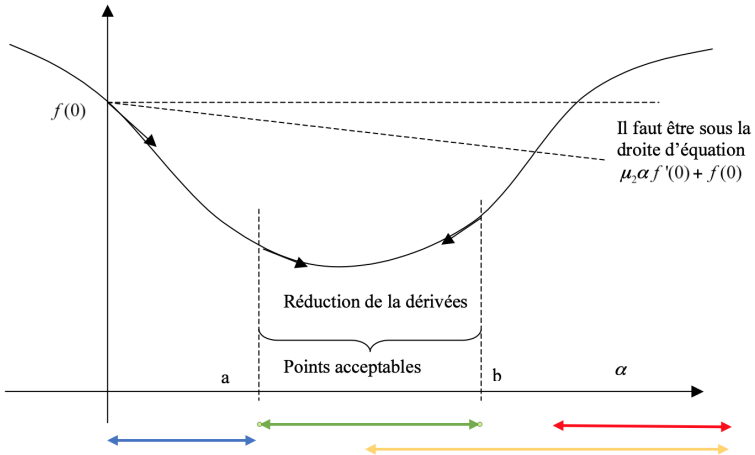
Algorithme de descente

Notation : $f(\alpha) = f(x_k + \alpha d_k)$

Algorithme (Fletcher) : $\alpha_0 = 0, \alpha_1 = 1$

Phase 1(Intervalle acceptable) :

Pour $i = 1, 2, \ldots$ faire



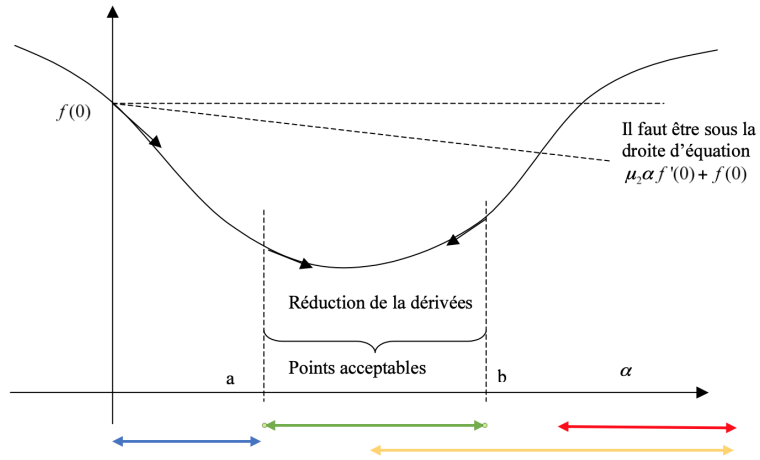
1. évaluer $f(\alpha_i)$
2. si $f(\alpha_i) > f(0) + \mu_2 \alpha_i f'(0)$ ou $f(\alpha_i) \geq f(\alpha_{i-1})$ zone
   alors $a_i = \alpha_{i-1}$ $b_i = \alpha_i$ fin de phase 1
   sinon évaluer $f'(\alpha_i)$
   si $\left|f'(\alpha_i)\right| \leq \mu_3 |f'(0)|$ alors fin de phase 1 et 2 et $\alpha = \alpha_i$ zone
   sinon
   si $f'(\alpha_i) \geq 0$ zone alors $a_i = \alpha_i$, $b_i = \alpha_{i-1}$ fin de phase 1
   sinon choisir $\alpha_{i+1} \in [\alpha_i + (\alpha_i - \alpha_{i-1}), \alpha_i + \tau_1(\alpha_i - \alpha_{i-1})]$zone

Fin Pour

Résultat : $I = [a_i, b_i]$

Dans la pratique, on peut choisir $\tau_1 = 9$.

Phase 2 (Réduction de l'intervalle $I = [a_i, b_i]$) :



Pour $j = i, i+1, \ldots$ faire

1. choisir $\alpha_j \in [a_j + \tau_2(b_j - a_j), b_j - \tau_3(b_j - a_j)]$
2. évaluer $f(\alpha_j)$
3. si $f(\alpha_j) > f(0) + \mu_2 \alpha_j f'(0)$ ou $f(\alpha_j) \geq f(a_j)$ zone
   alors $a_{j+1} = a_j$, $b_{j+1} = \alpha_j$
   sinon évaluer $f'(\alpha_j)$
   si $\left|f'(\alpha_j)\right| \leq -\mu_3 f'(0)$ zone alors fin de phase 2 et $\alpha = \alpha_j$
   sinon $a_{j+1} = \alpha_j$
   si $(b_j - a_j) f'(\alpha_j) \geq 0$ alors $b_{j+1} = a_j$ sinon $b_{j+1} = b_j$

Fin Pour

Résultat : Le pas est $\alpha = \alpha_j$

Dans la pratique, on peut choisir $\tau_1 = 9$ $\tau_2 = 0.1$ $\tau_3 = 0.5$.

Remarque : Il n'y a pas de relation d'ordre entre les a_i et b_i.

Choix d'un point sur [a b] par interpolation polynomiale

Comment choisir un α dans un intervalle [a b] ?

Interpollation cubique : (on sait exprimer la dérivée de f)
Sur [a b], si on dispose de $f_a = f(a)$, $f_b = f(b)$, $f_a^{'} = f^{'}(a)$ et $f_b^{'} = f^{'}(b)$, il existe un unique polynôme P de degré trois tel que :

$$P(a) = f_a \quad P^{'}(a) = f_a^{'}$$
$$P(b) = f_b \quad P^{'}(b) = f_b^{'}$$

Le minimum de $P(\alpha)$ est alors donné par

$$\alpha = b - (b-a)\frac{f_b^{'} + \beta_2 - \beta_1}{f_b^{'} - f_a^{'} + 2\beta_2}$$

avec

$$\beta_1 = f_a^{'} + f_b^{'} - 3\frac{f_a - f_b}{a-b}$$
$$\text{et } \beta_2 = \sqrt{\beta_1^2 - f_a^{'} f_b^{'}}$$

Choix du point sur [a b] par interpolation polynomiale

Interpollation quadratique : (on ne sait pas exprimer la dérivée de f)
Pour une interpolation quadratique sans calcul de g, on cherche un polynôme d'ordre 2. Il faut trois points (x_1, x_2, x_3), on peux choisir $x_1 = a$, $x_2 = (a+b)/2$ et $x_3 = b$, l'extremum est donné par :

$$\alpha = \frac{1}{2} \frac{\beta_{23} f_1 + \beta_{31} f_2 + \beta_{12} f_3}{\gamma_{23} f_1 + \gamma_{31} f_2 + \gamma_{12} f_3}$$

avec

$$\beta_{ij} = x_i^2 - x_j^2$$
$$\gamma_{ij} = x_i - x_j$$

La méthode du gradient

Nous disposons à présent **d'une méthode efficace pour déterminer le pas α_k à chaque itération de l'algorithme général.** Maintenant il nous faut choisir une direction de descente d_k.

La méthode du gradient

Idée : La direction de descente est celle de plus grande pente ; soit

$$d_k = -g_k.$$

La méthode converge globalement et est donc robuste mais avec en général de piètres performances (phénomène de zig zag).

Calcul de g_k : Si on ne dispose pas de $g(x_k)$, on évalue le taux d'accroissement de f dans chaque direction $e_i = [0 \ \cdots 0 \ 1 \ 0 \cdots \ 0]^T$ i.e.
Pour $i = 1, ..., n$:

$$g_i(x_k) = \frac{\partial f(x_k)}{\partial x_i} \approx \frac{\partial f(x_k + he_i) - f(x_k)}{h}$$

où g_i désigne la i-ième composante de g

Newton

Idée : On remplace f par son approximation quadratique au point x_k
Soit

$$q_k(d) = f_k + d^T g_k + \frac{1}{2} d^T H_k d$$

et on minime q_k plutôt que f.

La C.N. donne $H_k d_k = -g_k$ (Exercice)
Soit

$$d_k = -H_k^{-1} g_k.$$

q_k a un unique minimum si $H_k > 0$
L'itération est alors obtenue directement par

$$x_{k+1} = x_k + d_k \ (\alpha_k = 1)$$

Theorem

Si f est C^2 et si le Hessien H est défini positif et lipschitz dans un voisinage de la solution x^* alors la méthode converge dans un voisinage de la solution.
La convergence est quadratique près de la solution i.e.

$$\exists \beta > 0, \lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < \beta$$

Ceci signifie que la méthode améliore à chaque itération de deux chiffres significatifs l'estimation de la solution (prés de la solution).
Inconvénients : il faut évaluer et inverser à chaque itération le Hessien. La méthode peut ne pas converger si H_k n'est pas défini positif.
En résumé : la méthode est coûteuse, peu robuste mais rapide si elle converge.

Les Quasi-Newton

Calculer H numériquement et l'inverser est très coûteux.
Idée : Utiliser les valeurs calculées de f_k et g_k pour évaluer H_k^{-1}.
On pose :

$$s_k = x_{k+1} - x_k$$

$$p_k = g_{k+1} - g_k$$

Par un développement limité, on a

$$p_k = H_k s_k + o(s_k)$$

Pour une fonction quadratique, on a même l'égalité : $p_k = H_k s_k$
On pose alors la condition (**Quasi Newton**) suivante, on cherche une matrice noté B_{k+1} telle que

$$s_k = B_{k+1} p_k = H_k^{-1} p_k$$

Autrement dit, l'image par B_{k+1}de l'accroissement de gradient doit correspondre à l'accroissement de la position (comme pour H_k^{-1}).

La méthode BFGS (Broyden, Fletcher, Goldfard et Shanno)

$$B_{k+1} = B_k + (1 + \frac{p_k^T B_k p_k}{s_k^T p_k}) \frac{s_k s_k^T}{s_k^T p_k} - \frac{s_k p_k^T B_k + B_k p_k s_k^T}{s_k^T p_k}$$

$$B_0 = Id$$

La méthode DFP (Davidson, Fletcher et Powell)

$$B_{k+1} = B_k + \frac{s_k s_k^T}{s_k^T p_k} - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k}$$

$$B_0 = Id$$

Algorithme :
Pour $k = 0, 1, ...$

1. Poser $d_k = -B_k g_k$
2. Déterminer α_k par une procédure de descente dans la direction d_k et poser : $x_{k+1} = x_k + \alpha_k d_k$
3. Calculer B_{k+1} suivant DFP ou BFGS.

Fin Pour

Les Quasi -Newton

Proposition

Si la matrice B_k est maintenue définie positive alors d_k est toujours une direction de descente.

Preuve : Comme $d_k = -B_k g_k$, la condition de descente est vérifiée : $g_k^T d_k = -g_k^T B_k g_k < 0$

Proposition

$B_{k+1} > 0$ *si* $B_k > 0$ *et si* $p_k^T s_k > 0$

Commentaire : La condition $p_k^T s_k > 0$ n'est pas très exigeante car :

$$p_k^T s_k = \alpha_k (d_k^T g_{k+1} - d_k^T g_k)$$

Or comme d_k est une direction de descente, $-d_k^T g_k > 0$ et $\alpha_k > 0$, seul le terme $d_k^T g_{k+1}$ peut être négatif. Mais si on utilise une procédure de descente qui respecte **la condition iii. (Wolfe) du critère de convergence** alors comme

$$\left| d_k^T g_{k+1} \right| < \mu_3 \left| d_k^T g_k \right| , \mu_3 < 1$$

on a la propriété. On voit donc que contrairement à l'algorithme de Newton, on a la garantie de converger vers un point stationnaire. Cette garantie se paye en contrepartie par une vitesse de convergence plus faible.

Proposition

La vitesse de convergence de la méthode est superlinéaire i.e.

$$\exists \beta > 0, \lim \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = \beta < 1 \text{ (Convergence linéaire)}$$

$$\text{Si } \beta = 0, \text{ la convergence est dite super linéaire}$$

En pratique, les deux méthodes donnent de très bons résultats et sont plus efficaces que la méthode du gradient à pas optimal.

Les méthodes de type moindres carrés.

- Forme : Minimiser une norme au carré

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|F(x)\|^2 = \frac{1}{2}\sum_{i=1}^{N} F_i^2(x)$$

- Exemple : Estimation paramétrique (Identification)
  Faire "correspondre" un modèle avec des données.
  On cherche à déterminer p dans la relation

$$y = f(x, p)$$

  tel que

$$\min_{p} \sum_{i=1}^{N} (y(x_i, p) - y_{mes}(x_i)))^2$$

  avec $y_{mes}(x_i)$ la valeur expérimentale de y au point x_i.

- Idée : Exploiter la structure du problème

Les méthodes de type moindres carrés.

Si on note J la matrice Jacobienne de F
H_i la matrice Hessienne de chaque composante F_i alors

$$\begin{aligned} g(x) &= J^T(x)F(x) \\ H(x) &= J^T(x)J(x) + Q(x) \\ \text{avec } Q(x) &= \sum_i F_i(x)H_i(x) \end{aligned}$$

Idée : La matrice $Q(x) \to 0$ lorsque $\|F(x)\| \to 0$ ($F_i(x) \to 0$ pour x proche de la solution)

La méthode de Gauss Newton

Cette méthode détermine une direction d_k de descente en négligeant le terme $Q(x)$ par la méthode de Newton.
Autrement dit, on fait l'approximation $H \approx J^T J$ et on utilise Newton ce qui donne :

1. On résout
$$J_k^T J_k d_k = -J_k^T F_k$$
2. On utilise une procédure de descente pour obtenir
$$\alpha_k = \arg\min_{\alpha} f(x_k + \alpha d_k)$$

La méthode de Levenberg Marquard

On modifie la méthode de G.N. en ajoutant

$$(J_k^T J_k + \lambda_k Id) d_k = -J_k^T F_k$$

où le scalaire λ_k permet de contrôler la longueur et la direction de d_k :

- pour $\lambda_k = 0$, on retrouve G.N.
- pour $\lambda_k \to +\infty$, $\|d_k\| \to 0$ et $d_k \approx -\frac{1}{\lambda_k} J_k^T F_k$

  C'est l'algorithme du gradient et $\|F_{k+1}\| < \|F_k\|$ pour λ_k suffisamment grand.

  On gagne en robustesse au détriment de la vitesse.

  Implémentation :
  - L'inversion de $J_k^T J_k$ est obtenue par une factorisation QR (R triangulaire supérieure $Q^{-1} = Q^T$)
  - Le solveur doit contrôler efficacement λ_k.

Les méthodes de gradients conjugués

Idée : Utiliser une direction de descente conjuguée aux directions précédentes. Pour une fonction quadratique, l'algorithme converge en au plus n coups.

Les méthodes de gradients conjugués sont souvent employées pour des problèmes de grandes tailles car peu coûteuses.

Algorithmes :

$$d_1 = -g_1$$

Pour $k \geq 1$,

$$d_{k+1} = -g_{k+1} + \beta_k d_k$$

avec (**Algorithme de Fletcher-Reeves**) :

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$$

ou (**Algorithme de Polak-Ribière**) :

$$\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{g_k^T g_k}$$

Optimisation avec contraintes

Formulation du problème :
Trouver le minimum de la fonction f

$$\min_x f(x), x \in \mathbb{R}^n$$

sous les contraintes

$$c_i(x) = 0, i \in E = \{1, ..., m_e\}$$
$$c_i(x) \geq 0, i \in I = \{m_e + 1, ..., m\}$$

où $f : \mathbb{R}^n \to \mathbb{R}$
$c_i : \mathbb{R}^n \to \mathbb{R}$
On supposera que les fonctions c_i et f sont de classes C^2.

Problèmes avec contraintes

- **Points admissibles :** On notera D l'ensemble des points x qui satisfont les contraintes c.
- **Directions admissibles :** Soient $x \in D$ et une suite $x_k (\in D) \to x$.
  $\delta_k = x_k - x$, la suite des déplacements admissibles
  On note

$$\delta_k = \alpha_k s_k$$

  avec $s_k = \frac{\delta_k}{\|\delta_k\|}$, $\|s_k\| = 1$ et $\alpha_k = \|\delta_k\|$.

  Si $x_k \to x$ alors $\alpha_k \to 0$ et $s_k \to s$. La limite s est appelée une direction admissible au point x
- On notera $\mathcal{F}(x) = \{$ ensemble des directions admissibles au point $x\}$.

Conditions du premier ordre

Cas des contraintes égalités :

- Chaque contrainte égalité définie une hypersurface $S_i = c_i^{-1}(0)$ et

$$D = \cap_{i \in E} S_i.$$

- Notons $a_i(x) = \nabla c_i(x)$ et par $\mathcal{E}$, l'espace engendré par les $a_i(x)$, $i \in E$.
- $\Pi(x) = \mathcal{E}^{\perp} = \{z : z \perp a_i(x),\ i \in E\}$ est le plan tangent à D au point x.

- Montrons que les directions admissibles au point $x \in D$ appartiennent à $\Pi(x)$.
  Soient $x \in D$ et $x_k = x + \alpha_k s_k$ une suite de points admissibles avec $\alpha_k \to 0$ et $s_k \to s$

$$0 = \frac{c_i(x + \alpha_k s_k) - c_i(x)}{\alpha_k} = s_k^T a_i + o(\alpha_k)/\alpha_k$$

  Par passage à la limite,

$$c_i'(x; s) = s^T a_i(x) = 0, \forall i \in E, \forall s \in \mathcal{F}(x)$$

- Autrement dit, les directions admissibles s sont orthogonales à $a_i(x)$, $i \in E$ et donc appartiennent à $\Pi(x)$.
- Réciproquement on peut montrer que tout vecteur normé $v \in \Pi(x)$ est une direction admissible et conclure que $\mathcal{F}(x) = \Pi(x)$.

Conditions du premier ordre :

Cas des contraintes égalités :

Par un développement limité, si x^* un minimum local, la pente de f dans la direction s ne peut être négative et on doit donc avoir

$$f'(x^*; s) = s^T g^* \geq 0$$

Puisque $c_i \in C^1$, $-s \in \Pi(x^*)$ et il vient $s^T g^* = 0$ (puisque $\pm s^T g^* \geq 0$) pour tout $s \in \Pi(x^*)$.

Donc $g^* \in \Pi(x^*)^\perp = (\mathcal{E}^\perp)^\perp = \mathcal{E}$

Et g^* s'écrit nécessaire comme une combinaison linéaire des a_i^* :

$$g^* = \sum_{i \in E} \lambda_i^* a_i^* = A^* \lambda^*$$

avec $A^* = [a_1^* | a_2^* | \cdots | a_m^*]$ et $m = |E|$.

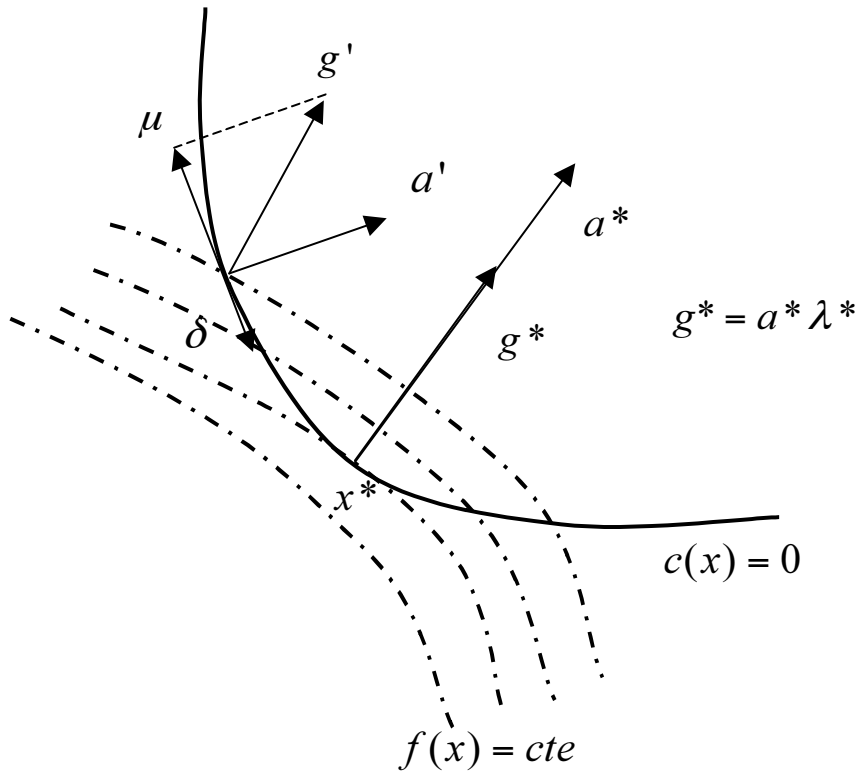


FIGURE: $g^* = A^* \lambda^*$

Conditions du premier ordre :

Cas des contraintes égalités

On définit alors **la fonction Lagrangienne**

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in E} \lambda_i c_i(x)$$

A l'optimum, il vient : $\nabla \mathcal{L}(x^*, \lambda^*) = 0$ i.e.

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= g^* - \sum_{i \in E} \lambda_i^* a_i^* \\ &= g^* - A^* \lambda^* \\ &= 0 \\ \nabla_\lambda \mathcal{L}(x^*, \lambda^*) &= c_i^* = 0, \ i \in E \end{aligned}$$

Conditions du premier ordre

Cas général :

- On note : $\mathcal{A}^* = E \cup I^*$ où I^* est le sous ensemble de I des contraintes inégalités actives à l'optimum i.e

$$c_i^* = 0, \forall\ i \in \mathcal{A}^*$$

- Puisque $c_i^* = 0$ et $c_i(x^* + \delta) \geq 0$ pour $i \in I^*$ et δ admissible, les directions admissibles s doivent vérifier

$$c_i'(x^*; s) = s^T a_i^* = 0, \forall i \in E$$
$$c_i'(x^*; s) = s^T a_i^* \geq 0, \forall i \in I^*$$
$$\text{et } f'(x^*; s) = s^T g^* \geq 0$$

Conditions du premier ordre

Cas général :

- On montre que le gradient de f s'exprime comme une combinaison linéaire des gradients des contraintes actives

$$g^* = \sum_{i \in E \cup I^*} \lambda_i^* a_i^*$$

avec les $\lambda_i^* \geq 0$, $i \in I^*$.

- Sinon (cas d'une seule contrainte inégalité) $\exists \lambda < 0$ et on peut toujours choisir s tel que (CN)

$$s^T a^* > 0$$

d'où

$$s^T g^* = s^T a^* \lambda < 0$$

d'où la contradiction.

Hypothèse de régularité

Definition

On note

$$F(x^*) = \{s : s \neq 0 \text{ et } s^T a_i^* = 0,\ \forall i \in E,\ s^T a_i^* \geq 0,\ \forall i \in I^*\}$$

On a de manière évidente $F^* \supset \mathcal{F}^*$, réciproquement on a le

Lemma

$F^* = \mathcal{F}^*$

si les contraintes sont linéaires ou

si les a_i^* sont linéairement indépendantes au point x^*

Conditions du premier ordre

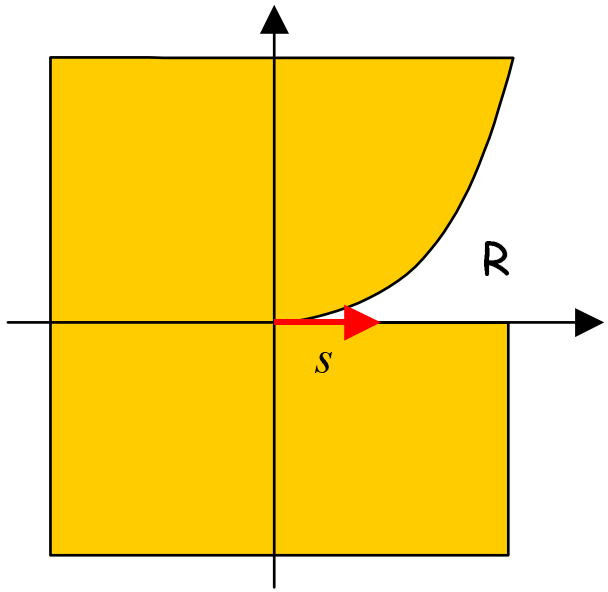
Contraintes :

$$c_1(x) = x_1^3 - x_2 \geq 0$$

$$c_2(x) = x_2 \geq 0$$

gradients : $\begin{cases} a_1(x) = \begin{pmatrix} 3x_1^2 \\ -1 \end{pmatrix} \\ a_2(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{cases}$

au point $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow s = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$



Contraintes inégalités : Cas où $-s \notin \mathcal{F}(x)$ et où $-s \in F(x)$

Conditions Nécessaires Cas général :

On considère : $\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in E \cup I} \lambda_i c_i(x)$

Theorem (Kuhn-Tucker (KT) Conditions)

Sous l'hypothèse de régularité $F^* = \mathcal{F}^*$ et si x^* est un minimiseur local du problème avec contraintes, alors il existe un multiplieur de Lagrange λ^* tel que

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = g^* - \sum_{i \in E \cup I} \lambda_i^* a_i^* = 0$$

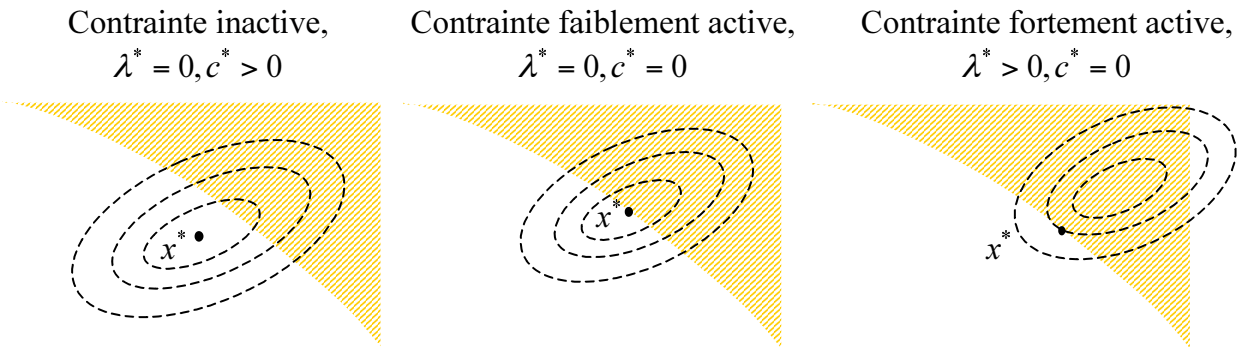
$$c_i^* = 0, \ i \in E$$

$$c_i^* \geq 0, \ i \in I$$

$$\lambda_i^* \geq 0, \ i \in I$$

$$\lambda_i^* c_i^* = 0, \forall i$$

Remarque : $\lambda_i^* = 0$ *si* $c_i^* > 0$.



Optimum et contrainte inégalité

Remarque concernant la signification de $\lambda_i^* \geq 0$.

Trouver min f sous une contrainte inégalité : $c(x) \geq 0$.

- Remplaçons cette contrainte par $c(x, \varepsilon) = c(x) - \varepsilon \geq 0$ pour un $\varepsilon > 0$. On note par $f^*(\varepsilon)$ la valeur de f à l'optimum en fonction de ε sous la contrainte $c(x, \varepsilon) = 0$.
- Pour tout $\varepsilon > 0$, $f^*(\varepsilon) \geq f^*(0)$ (restriction de D) et donc que

$$\left.\frac{df^*(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} \geq 0$$

- On peut écrire que $f^*(\varepsilon) = \mathcal{L}(x^*(\varepsilon), \lambda^*(\varepsilon), \varepsilon)$ et par dérivation il vient :

$$\frac{df^*(\varepsilon)}{d\varepsilon} = \nabla_x \mathcal{L}(x^*(\varepsilon), \lambda^*(\varepsilon), \varepsilon)\frac{\partial x}{d\varepsilon} + \nabla_\lambda \mathcal{L}(x^*(\varepsilon), \lambda^*(\varepsilon), \varepsilon)\frac{\partial \lambda}{d\varepsilon} + \frac{\partial \mathcal{L}}{\partial \varepsilon}$$

$$\left.\frac{df^*(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} = \left.\frac{\partial \mathcal{L}}{\partial \varepsilon}\right|_{\varepsilon=0} = \lambda^* \geq 0$$

- Autrement dit le multiplieur de Lagrange correspond à la dérivé de $f^*(\varepsilon)$. Il indique qu'un déplacement vers l'intérieur du domaine admissible D augmente le critère f^*. C'est donc bien une condition nécessaire de minimum.

Conditions d'ordre 2

Cas des contraintes égalités :
Si x^* est un minimum local alors pour un déplacement admissible δ, on a

$$\begin{aligned} f(x^* + \delta) &= \mathcal{L}(x^* + \delta, \lambda^*) \\ &= \mathcal{L}(x^*, \lambda^*) + \delta^T \nabla_x \mathcal{L}(x^*, \lambda^*) + \frac{1}{2}\delta^T W^* \delta + o(\delta^T \delta) \\ &= f(x^*) + \frac{1}{2}\delta^T W^* \delta + o(\delta^T \delta) \end{aligned}$$

avec $W^* = \nabla_x^2 \mathcal{L}(x^*, \lambda^*) = \nabla_x^2 f(x^*) - \sum_i \lambda_i^* \nabla_x^2 c_i(x^*)$.

En passant à la limite, si $f(x^*)$ est un minimum alors on doit avoir $s^T W^* s \geq 0$ pour toutes les directions admissibles. Ceci détermine une condition nécessaire.

Cas général :

La condition suffisante vient de :

Theorem (**Condition Suffisante de minimum local**)

Si (x^*, λ^*) *satisfait les conditions d'ordre 1 (KT cond) et si*

$$s^T W^* s > 0, \ \forall s \in G^*$$

où

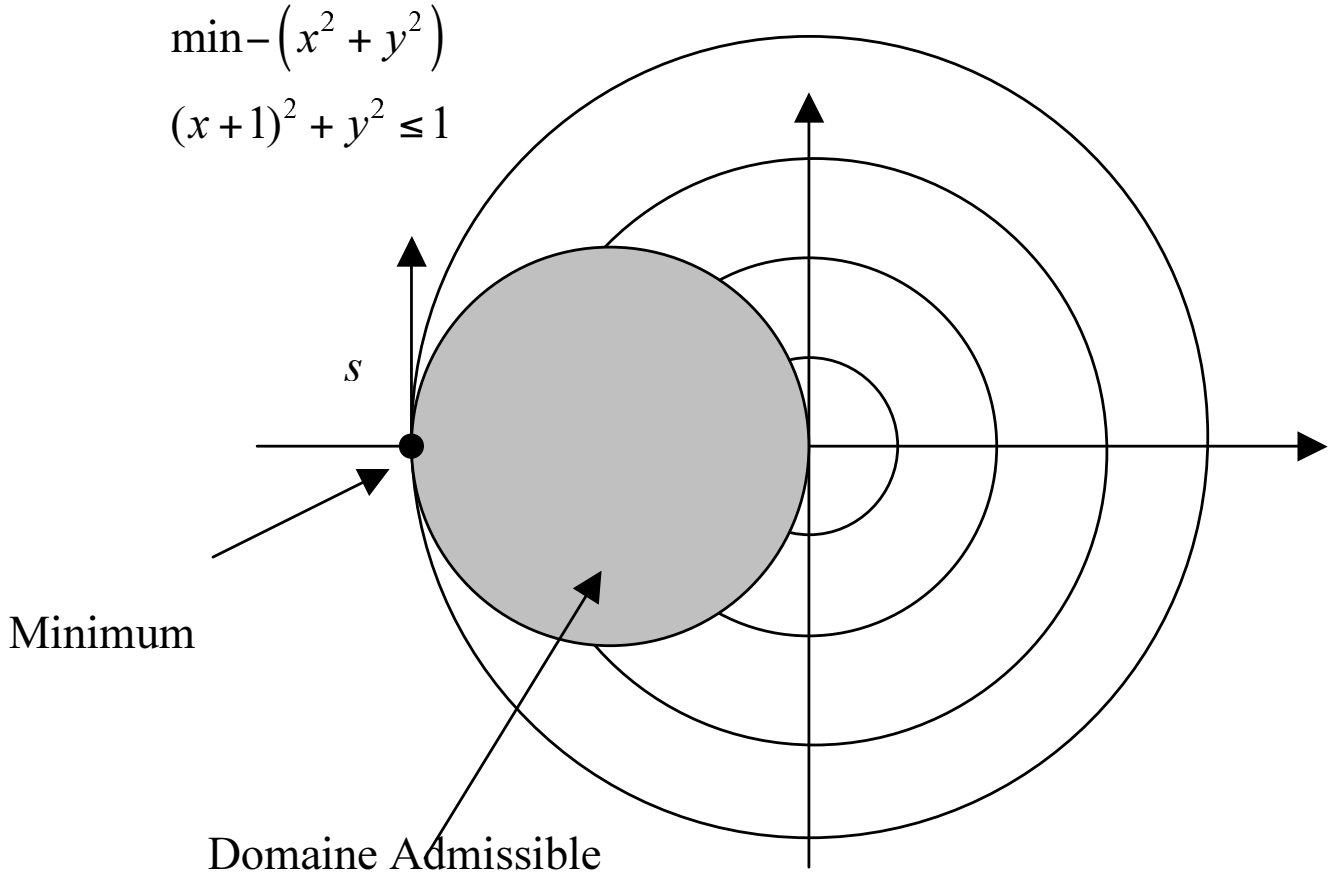
$$G^* = \{s : s \neq 0, \ a_i^{*T} s = 0, \ i \in \mathcal{A}_+^*, \ a_i^{*T} s \geq 0, \ i \in \mathcal{A}^* \backslash \mathcal{A}_+^*\}$$

et

$$\mathcal{A}_+^* = \{i : i \in E \text{ ou } \lambda_i^* > 0\} \text{ contrainte fortement active}$$

alors x^* *est un minimiseur local.*

Attention : La condition porte bien sur le Hessien du Lagrangien W et non sur le Hessien de f comme en témoigne l'exemple de la figure



Exemple où $s^T \nabla^2 f s < 0$ à l'optimum (courbure négative)

Exemple

Résoudre

$$\min_x f(x) = x_1^2 - x_1 x_2 + x_2^2 - 3x_1$$

sous la contrainte :

$$2 - x_1 - x_2 \geq 0$$

Problème Linéaire Standard :
Trouver min $c^T x$ sous la contrainte $Ax = b$, $x \geq 0$ avec $A_{m \times n}$, $m \leq n$

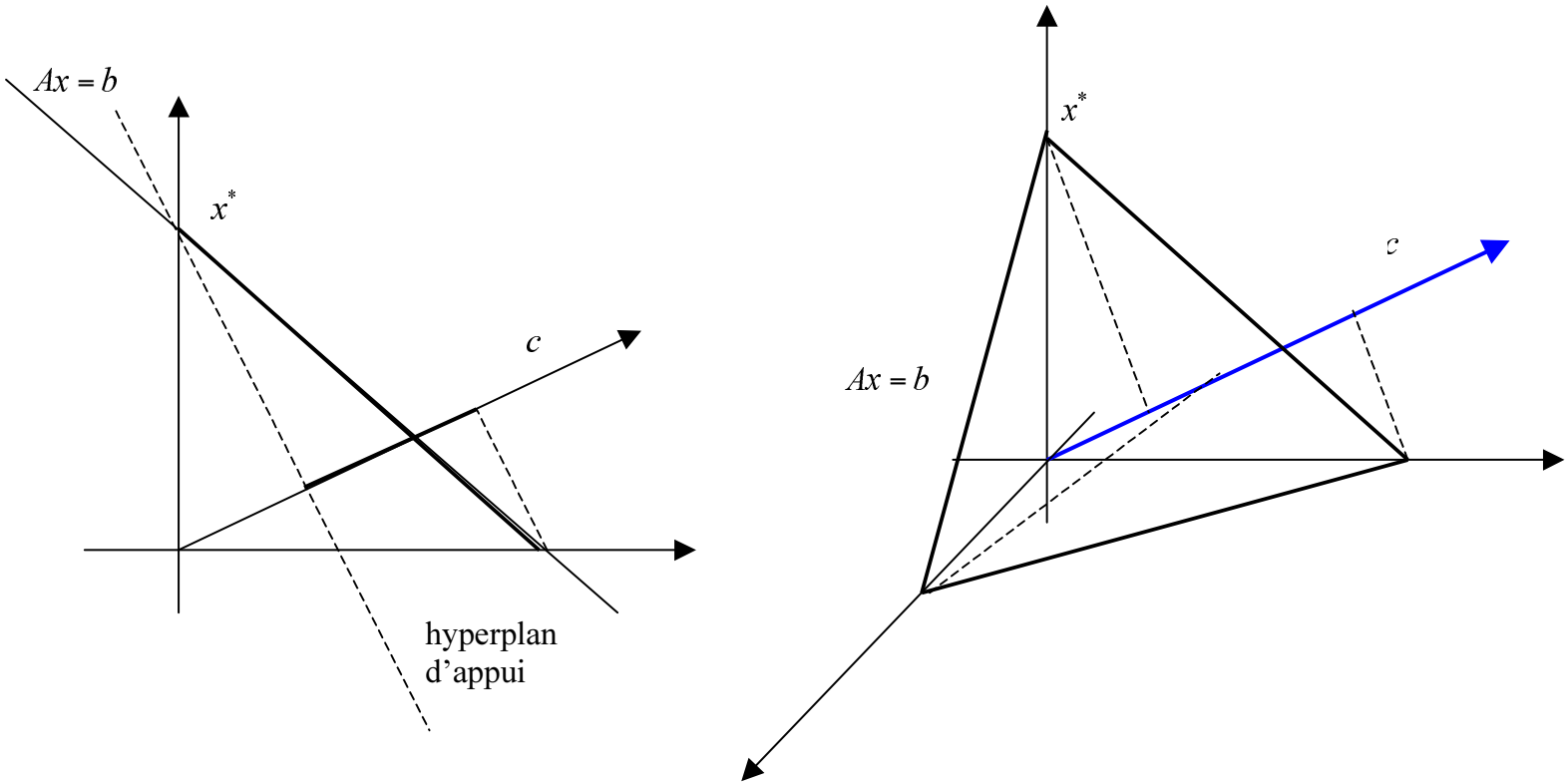
Passage sous la forme standard :

- min $c^T x$ sous la contrainte $Ax \leq b$, $x \geq 0$.
  On considère une variable additionnelle z, on pose $Ax + z = b$, $z \geq 0$ et on retrouve la forme standard mais pour le vecteur (x, z).
- min $c^T x$ sous la contrainte $Ax \leq b$
  On considère une variable additionnelle z, on pose $Ax + z = b$, $z \geq 0$ et on décompose $x = u - v$ avec $u \geq 0$ $v \geq 0$ et on retrouve la forme standard avec (u, v, z) :

$$Au - Av + z = b, \ u \geq 0, \ v \geq 0, \ z \geq 0$$

Propriété : L'ensemble des points admissibles est un polyèdre convexe. La solution se situe sur un sommet.

La programmation linéaire



Idée : Exploiter le fait qu'il y a $n - m$ composantes de la solution x^* nulles. Il faut éviter de tester tous les sommets car C_n^{n-m} est en général très grand.

Effectuer des déplacements de sommet en sommet en empruntant les arrêtes du polyèdre

Algorithme du simplexe

On part d'un sommet x du polyèdre (x a $(n-m)$ composantes nulles).
On pose (à une permutation près)

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$$

avec x_B : variable "basic" et x_N : variable "nulle".

$$A = [A_B | A_N]$$

où A_B est non singulière si A est de plein rang.

Il vient

$$Ax = A_B x_B + A_N x_N = b$$

d'où

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} A_B^{-1} b \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{b} \\ 0 \end{bmatrix}$$

et la contrainte $x_i > 0$ donne $\hat{b} > 0$.
La fonction coût est alors réduite à :

$$f = c^T x = c_B^T x_B \text{ avec } c = \begin{bmatrix} c_B \\ c_N \end{bmatrix}.$$

Algorithme du simplexe

On cherche à présent à évaluer l'influence d'une perturbation de x_N sur f.

Si x_N est non nul, on a

$$x_B = A_B^{-1}(b - A_N x_N) = \hat{b} - A_B^{-1} A_N x_N,$$

d'où

$$\begin{aligned} f(x_N) &= c_B^T(\hat{b} - A_B^{-1} A_N x_N) + c_N^T x_N \\ &= \hat{f} + \hat{c}^T x_N \text{ avec } \hat{c} = c_N - A_N^T A_B^{-T} c_B \end{aligned}$$

Comme $x_N \geq 0$, on voit que x est optimal si

$$\hat{c} \geq 0 \tag{1}$$

Sinon on choisit la composante $\hat{c}_q$ la plus négative.
Soit

$$\hat{c}_q = \min_i \hat{c}_i, \hat{c}_i < 0$$

et en augmentant x_{N_q}, on fait baisser le critère (les autres composantes de x_N restent à zéro) tout en satisfaisant la contrainte $Ax = b$.
Mais attention la valeur de x_{N_q} est également limitée par la contrainte $x \geq 0$.

Algorithme du simplexe

Regardons l'effet de x_{N_q} sur x_B :
Comme $x_B = \hat{b} - A_B^{-1} A_N x_N$ et $x_N = (0, \cdots, 0, x_{N_q}, 0, \cdots, 0)$ il vient :

$$\begin{aligned} x_B &= \hat{b} - A_B^{-1} a_q x_{N_q} \\ &= \hat{b} + d x_{N_q} \end{aligned}$$

avec a_q la q ième colonne de A_N et $d = -A_B^{-1} a_q$

On observe que si x_{N_q} augmente et si $d_i < 0$ alors x_{B_i} diminue. La première composante de x_B a atteindre la contrainte $x_i = 0$ est alors obtenue pour un indice p tel que :

$$x_{N_q} = \frac{\hat{b}_p}{-d_p} = \min_{\substack{i \in B \\ d_i < 0}} \frac{\hat{b}_i}{-d_i}$$

Pour cette valeur de x_{N_q}, la contrainte $x_{B_p} = 0$ devient active.
Bilan :

- la valeur du critère a baissée,
- $x_{N_q} > 0$ et devient une variable "Basic"
- $x_{B_p} = 0$ et devient une variable "Nulles"

On recommence en permutant les éléments de la colonne p avec la colonne q.
Et ainsi de suite jusqu'à l'obtention de la condition nécessaire $\hat{c} \geq 0$.

Algorithme du simplexe

On part d'un sommet x du polyèdre. On note $x = (x_B, x_N)$ ($\equiv$ permutation).
Sous la forme d'un tableau cela donne :

c_B^T	c_N^T	0
A_B	A_N	b

Puis par la méthode d'élimination de Gauss

0	$\hat{c}^T$	$-\hat{f}$
Id	$\hat{A}_N = A_B^{-1} A_N$	$\hat{b}$

1. On choisit alors la colonne correspondant au terme $\hat{c}_q$ le plus négatif. Soit a_q la partie de la colonne correspondant à $\hat{A}_N$.
2. Puis on sélectionne la ligne p pour laquelle le rapport $\frac{\hat{b}_p}{a_{q_p}}$ est minimum avec $a_{q_p} > 0$.
3. On utilise alors l'élément situé sur la ligne p et colonne q comme nouveau pivot de Gauss, (on peut permuter les colonnes correspondantes)
4. On revient au point 1 tant qu'il existe des valeurs de $\hat{c}_q$ négatives.

Algorithme du simplexe

Initialisation de l'algorithme : Choix du sommet initial du polyèdre

La solution du problème linéaire

$$\min \sum_i z_i \tag{2}$$

$$Ax + Dz = b \tag{3}$$

$$x \geq 0,\ z \geq 0 \tag{4}$$

avec D diagonale et $d_{ii} = 1$ si $b_i \geq 0$ et $d_{ii} = -1$ sinon, est un sommet du polyèdre.

Conclusion : On résout par l'algorithme du simplexe le problème ci-dessus en partant du point $(0, Db)$ qui est un point admissible.

Si la solution trouvée (x^*, z^*) est telle que $z^* \neq 0$, le problème initial n'a pas de solution, sinon on initialise le problème initial avec x^*.

Algorithme du simplexe

Example

A titre d'exemple, considérons le problème suivant :
Trouver

$$\min x_1 + 2x_2 + 3x_3 + 4x_4$$

sous la contrainte

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= 1 \\ x_1 + x_3 - 3x_4 &= 1/2 \\ x &\geq 0 \end{aligned}$$

Algorithme du simplexe

On part de $x_B = \{x_1, x_2\}$ $x_N = \{x_3, x_4\}$ (admissible)

$k = 0$

1	2	3	4	0
1	1	1	1	1
1	0	1	-3	$1/2$

$k = 1$

0	0	2	-1	$-3/2$
1	0	1	-3	$1/2$
0	1	0	4	$1/2$

Permutation $x_2,\ x_4 :\ x_B = \{x_1, x_4\}$ $x_N = \{x_2, x_3\}$

0	-1	2	0	$-3/2$
1	-3	1	0	$1/2$
0	4	0	1	$1/2$

Algorithme du simplexe

$k = 2$

0	0	2	1/4	−11/8
1	0	1	3/4	7/8
0	1	0	1/4	1/8

d'où la solution $x^* = (7/8, 0, 0, 1/8)$ et $f^* = 11/8$.

Algorithme du simplexe

Example

En utilisant une variable additionnelle $z = (z_1, z_2)$ (voir la remarque en début de section) déterminer

$$\min x_1 + 2x_2 + 3x_3 + 4x_4$$

sous la contrainte

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &\leq 1 \\ x_1 + x_3 - 3x_4 &\leq 1/2 \\ x &\geq 0 \end{aligned}$$

La programmation quadratique

Trouver

$$\min q(x) = \frac{1}{2} x^T H x + g^T x$$

sous la contrainte

$$a_i^T x = b_i, i \in E$$
$$a_i^T x \geq b_i, i \in I$$

avec H matrice symétrique et g un vecteur (constant).

Généralités :

- Si $H \geq 0$, $\exists$ un minimum global. ($\mathcal{R} \neq \varnothing$)
- Si $H > 0$, le minimum est unique

Cas où $I = \varnothing$

Trouver

$$\min q(x) = \frac{1}{2} x^T H x + g^T x$$

sous la contrainte

$$A^T x = b$$

avec $A = [a_1 \cdots a_m]$ et $m = |E|$

Hypothèses : $m < n$ et rang $A^T = m$
(les contraintes sont linéairement indépendantes)

Résolution directe (Les méthodes d'éliminations)

Idée : Transformer par substitution le problème initial en un problème sans contrainte

En posant $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ avec $x_1 \in \mathbb{R}^m$ et $x_2 \in \mathbb{R}^{n-m}$, $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, $g = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$, $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$, il vient

$$x_1 = A_1^{-T}(b - A_2^T x_2).$$

Par substitution dans $q(x)$, on est ramené à résoudre un problème quadratique sans contrainte :

$$\min_{x_2} \Psi(x_2) = \frac{1}{2} \begin{bmatrix} A_1^{-T}(b - A_2^T x_2) \\ x_2 \end{bmatrix}^T H \begin{bmatrix} A_1^{-T}(b - A_2^T x_2) \\ x_2 \end{bmatrix} + g^T \begin{bmatrix} A_1^{-T}(b - A_2^T x_2) \\ x_2 \end{bmatrix}$$

pour lequel la condition nécessaire $\frac{\partial \Psi(x_2^*)}{\partial x_2} = 0$ permet de déterminer l'unique solution. Exercice : déterminer x_2^*

Problème : A_1^T doit être inversible

Les méthodes d'éliminations généralisées

Posons $Y_{n\times m}$ et $Z_{n\times n-m}$ telles que $[Y|Z]$ est non singulière et

$$A^T Y = Id$$
$$A^T Z = 0$$

Z est une base de $Ker(A^T)$

Y^T est l'inverse généralisée à gauche de A ($Y^T A = Id$).

On voit alors que

$$x = Yb$$

est une solution de $A^T x = b$.
En fait, toutes les solutions s'écrivent :

$$x = Yb + \delta$$

avec $\delta \in Ker(A^T)$. Autrement dit, il existe y : $\delta = Zy$, $y \in \mathbb{R}^{n-m}$.

Les méthodes d'éliminations généralisées

En résumé, tout point admissible vérifie

$$x = Yb + Zy.$$

On remplace alors dans $q(x)$, x par $Yb + Zy$ d'où (Exercice)

$$\Psi(y) = \frac{1}{2} y^T Z^T H Z y + (g + HYb)^T Z y + \frac{1}{2}(g + HYb)^T Yb.$$

Si $Z^T HZ > 0$, il existe un unique y^* déterminé par $\nabla\Psi(y^*) = 0$. Soit

$$Z^T HZ y^* = -Z^T(g + HYb)$$

Par ailleurs, en appliquant les condition de KT, λ^* est obtenu par

$$\lambda^* = Y^T(Hx^* + g).$$

Preuve : $L(x, \lambda) = \frac{1}{2} x^T Hx + g^T x - \lambda^T(A^T x - b)$ et
$\nabla_x L^* = 0 : Hx^* + g - A\lambda^* = 0 \rightarrow Y^T(Hx^* + g) = \lambda^*$

Les choix de Y et Z peuvent être fait de différentes façons :
Par **une factorisation QR**

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R$$

avec $Q_{n\times n}$ orthogonal ($Q^{-1} = Q^T$) et R triangulaire supérieure.(Q_1 de dimension $n \times m$, Q_2 de dimension $n \times (n-m)$, R de dimension $m \times m$)
Alors

$$Y = Q_1 R^{-T}$$

et

$$Z = Q_2$$

La méthode des contraintes actives

Cas général $I \neq \varnothing$:

Idée : Résoudre une suite de problèmes quadratiques avec contraintes égalités et utiliser un mécanisme d'ajout-suppression de contraintes pour parvenir à la solution du problème initial.

Une contrainte inégalité $c(x) \geq 0$ est dite active au point x si $c(x) = 0$.

On note $\mathcal{A}(x) = \{i \in E \cup I : c_i(x) = 0\}$ l'ensemble des contraintes actives au point x.

La méthode des contraintes actives

Cas général

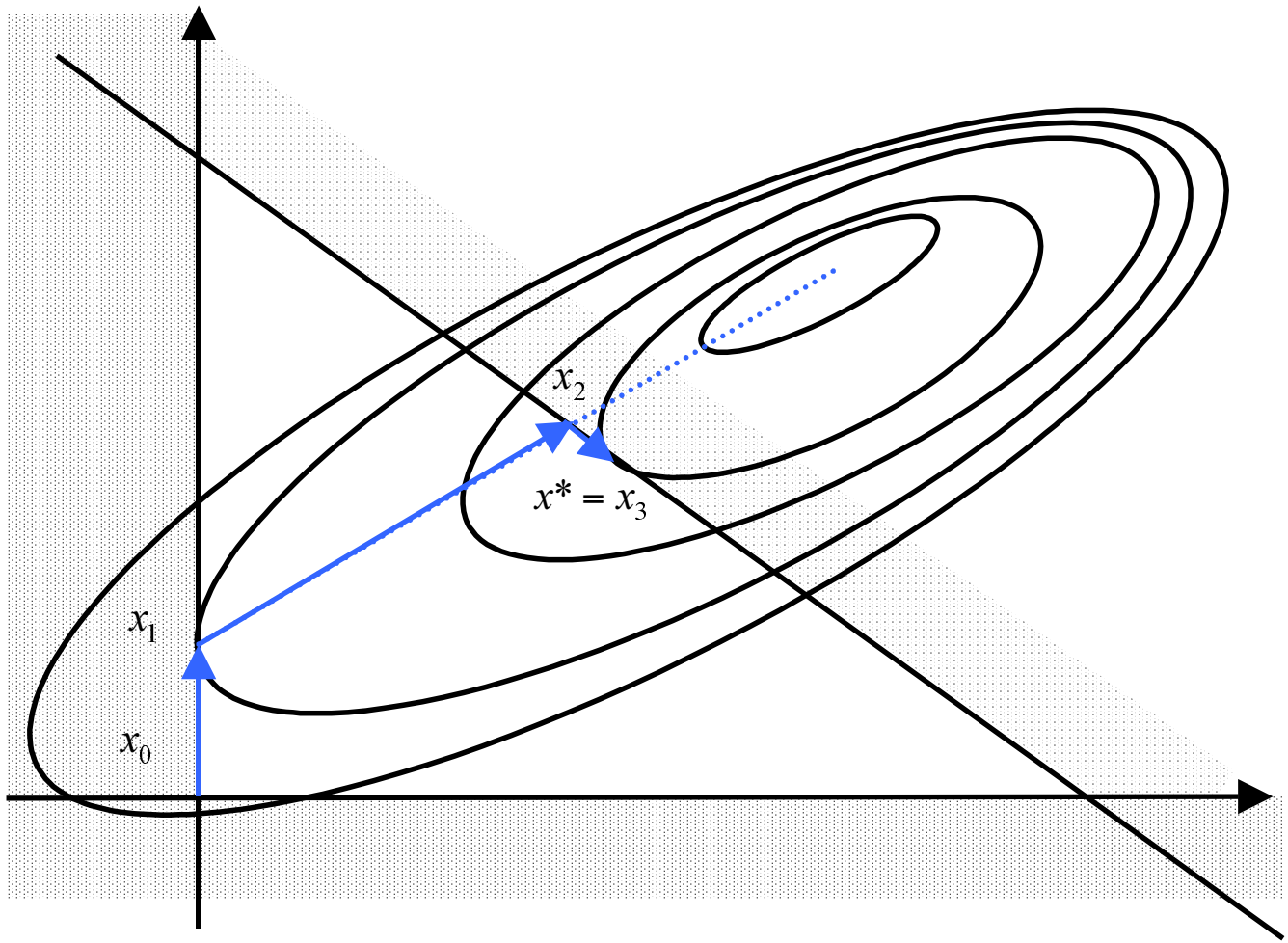


FIGURE: La méthode des contraintes actives

La méthode des contraintes actives

On applique l'algorithme suivant :
Pour k=0,1,...

1. A partir du point x_k admissible, on détermine $\mathcal{A}(x_k)$ et on résout le sous problème :

$$\min_x \frac{1}{2} x^T H x + g^T x \tag{5}$$
$$a_i^T x = b_i, \ \ i \in \mathcal{A}(x_k)$$

si on note $x = x_k + \delta$, ce problème est équivalent à (à faire en exercice) :

$$\min_\delta \frac{1}{2} \delta^T H \delta + g_k^T \delta \tag{6}$$
$$a_i^T \delta = 0, \ \ i \in \mathcal{A}(x_k)$$

avec $g_k = g + Hx_k$.
Notons δ_k la solution obtenue (par une méthode d'éliminations).

La méthode des contraintes actives

2. Si δ_k est admissible pour les contraintes $i \notin \mathcal{A}(x_k)$ alors

$$x_{k+1} = x_k + \delta_k$$

3. Sinon il existe un indice $i \notin \mathcal{A}(x_k)$ tel que :

$$a_i^T(x_k + \delta_k) < b_i, \ \ i \notin \mathcal{A}(x_k)$$

et il faut donc réduire le déplacement δ_k.
Comme $a_i^T x_k \geq b_i$ $(x_k \in D)$, on a : $a_i^T \delta_k < 0$ et on peut écrire :

$$1 > \frac{b_i - a_i^T x_k}{a_i^T \delta_k}, \text{ pour au moins un indice } i \notin \mathcal{A}(x_k)$$

On choisit alors de saturer le déplacement suivant :

$$x_{k+1} = x_k + \alpha_k \delta_k \text{ avec } \alpha_k = \min(1, \min_{\substack{i \notin \mathcal{A}(x_k) \\ a_i^T \delta_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T \delta_k})$$

Si on note $p = \arg \min_{\substack{i \notin \mathcal{A}(x_k) \\ a_i^T \delta_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T \delta_k}$ et si $\alpha_k < 1$ alors la contrainte p devient active au point x_{k+1}.

La méthode des contraintes actives

4. Si $\alpha_k < 1$, $\mathcal{A}(x_{k+1}) = \mathcal{A}(x_k) \cup \{p\}$ et on recommence en 1.
5. Sinon on calcule les multiplicateur de Lagrange λ_i, au point x_{k+1} pour $i \in \mathcal{A}(x_k) \cap I$.
   - Si $\lambda_i \geq 0$ pour tout $i \in \mathcal{A}(x_k) \cap I$, on arrête et $x^* = x_{k+1}$ (les C.N. sont satisfaites)
   - Sinon on pose $\mathcal{A}(x_{k+1}) = \mathcal{A}(x_k) \setminus \{q\}$ (enlève la contrainte) avec

$$q = \arg \min_{i \in \mathcal{A}(x_k) \cap I} \lambda_i$$

   et on recommence en 1.

Résumé : Soit x_0 donné admissible

Pour k=0,1,...

1. Résoudre le sous problème :

$$\min_{\delta} \frac{1}{2}\delta^T H\delta + g_k^T \delta$$
$$a_i^T \delta = 0, i \in \mathcal{A}(x_k)$$

avec $g_k = g + Hx_k$. Soit δ_k la solution.

Poser: $x_{k+1} = x_k + \alpha_k \delta_k$ avec $\alpha_k = \min \left(1, \min_{\substack{i \notin \mathcal{A}(x_k) \\ a_i^T \delta_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T \delta_k} \right)$

2. Si $\alpha_k < 1$, poser $\mathcal{A}(x_{k+1}) = \mathcal{A}(x_k) \cup \{p\}$ avec $p = \arg \min_{\substack{i \notin \mathcal{A}(x_k) \\ a_i^T \delta_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T \delta_k}$.

3. Sinon calculer λ_i, au point x_{k+1} pour $i \in \mathcal{A}(x_k) \cap I$.
   - Si $\lambda_i \geq 0$, $\forall i \in \mathcal{A}(x_k) \cap I$, sortir de la boucle avec $x^* = x_{k+1}$
   - Sinon poser $\mathcal{A}(x_{k+1}) = \mathcal{A}(x_k) \setminus \{q\}$ avec $q = \arg \min_{i \in \mathcal{A}(x_k) \cap I} \lambda_i < 0$.

fin pour

Algorithme des contraintes actives

Example

Mettre en oeuvre l'algorithme des contraintes actives en partant de $x_0 = (0.5, 0)$ pour déterminer

$$\min(x_1 - 1)^2 + (x_2 - 2)^2$$

sous la contrainte

$$x_1 + 2x_2 \leq 2$$
$$x \geq 0$$

Même chose en partant $x_0 = (0, 0.5)$

Les méthodes principales sont la pénalisation et **SQP** (Sequential Quadratic Programming).

La méthode de pénalisation, Cas où $I = \varnothing$

Idée : Résoudre une suite de problèmes sans contraintes dont les solutions conduisent à celle du problème originale avec contraintes.

On considère la fonction de pénalisation

$$\Phi(x, \sigma) = f(x) + \frac{1}{2}\sigma c^T(x)c(x)$$

avec σ un nombre positif qui déterminent le poids de la pénalisation.

On cherche :

$$x^*(\sigma) = \arg\min_x \Phi(x, \sigma), \quad \sigma \to +\infty$$

La méthode de pénalisation

Algorithme :

i. Choisir une séquence $\sigma_k \to +\infty$ par exemple $\{1,10,100,...\}$

ii. Pour chaque σ_k, trouver le minimum $x^*(\sigma_k)$

iii. Terminer lorsque $c(x^*(\sigma_k))$ est suffisamment petit

Avantage : Facile à implémenter

Inconvénient : Lorsque $\sigma \nearrow$, on obtient un mauvais conditionnement de $\nabla^2\Phi$ ce qui pose des problèmes de convergence.

En effet $\nabla^2\Phi_k = W_k + \sigma_k A_k A_k^T$ et comme $A_k^T A_k$ est de rang m, il existe m valeurs propres de $\nabla^2\Phi_k$ qui tendent vers $+\infty$ avec σ_k.

Conséquences : Les choix de σ_1 puis de la vitesse d'évolution de σ_k sont délicats et la méthode peut se révéler inefficace.

La méthode SQP (Sequential Quadratic Programming)

C'est le standard pour le cas général.
Idée : Remplacer f non linéaire par l'approximation quadratique au point x_k suivante :

$$q_k(d) = g_k^T d + \frac{1}{2} d^T W_k d$$

avec $W_k = \frac{\partial^2 \mathcal{L}}{\partial^2 x}$ et les contraintes par leur linéarisation au point x_k.
Soit :

$$\min_d q_k(d) = g_k^T d + \frac{1}{2} d^T W_k d \tag{7}$$
$$c_i(x_k) + a_i^T(x_k) d \geq 0, i \in I$$
$$c_i(x_k) + a_i^T(x_k) d = 0, i \in E$$

La CN donne (cas $I = \varnothing$)

$$W_k d + g_k = A_k \lambda$$
$$c_k + A_k^T d = 0$$

avec $A_k = [a_1(x_k) \cdots a_m(x_k)]$ et $m = |E|$

La méthode SQP (Sequential Quadratic Programming)

La méthode se justifie dans le cas $I = \varnothing$, en appliquant Newton sur le Lagrangien.

$$\nabla^2 \mathcal{L}_k \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = -\nabla \mathcal{L}_k$$

Soit

$$\begin{pmatrix} W_k & -A_k \\ -A_k^T & 0 \end{pmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = \begin{bmatrix} -g_k + A_k \lambda_k \\ c_k \end{bmatrix}.$$

En posant, $\lambda_{k+1} = \lambda_k + \delta\lambda$ et $d_k = \delta x$, on parvient à :

$$\begin{pmatrix} W_k & -A_k \\ -A_k^T & 0 \end{pmatrix} \begin{bmatrix} d_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -g_k \\ c_k \end{bmatrix}$$

c'est la CN que l'on obtient sur le problème posé ci-dessus.

La méthode SQP (Sequential Quadratic Programming)

Algorithme

Pour $k = 1, 2, 3, ...$

1. Résoudre le problème quadratique (7) pour déterminer d_k et λ_{k+1}
2. Poser $x_{k+1} = x_k + \alpha_k d_k$ où α_k est choisi pour obtenir une réduction suffisante sur une fonction de mérite (ou pénalisation)
   $\Psi(x) = f(x) + \sum_{i \in E} v_i \, |c_i(x)| - \sum_{i \in I} v_i \min(c_i(x), 0)$, $v_i > 0$.

Convergence : La convergence est assurée localement si (x^*, λ^*) satisfont les conditions de second ordre. Si le point x_0 est suffisamment proche de x^* et si λ_k reste suffisamment proche de λ^* alors la séquence x_k converge vers x^* avec une vitesse d'ordre 2.

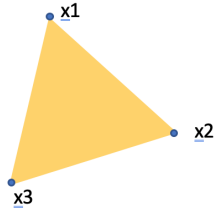
Il n'y a pas de garantie dans les autres cas.

Problèmes convexes

Definition (Ensemble Convexe)

K est convexe si $\forall x_0, x_1 \in K$, $\forall \theta \in [0\ 1]$, $\theta x_0 + (1- \theta)x_1 \in K$

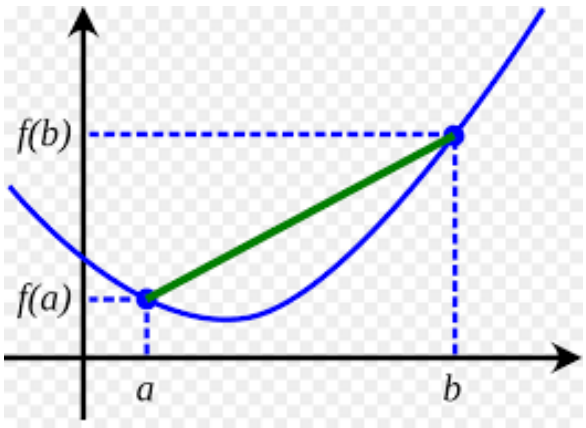
ou encore pour tout $x_i \in K$, $i = 1, ..., m$ alors tout x s'écrivant $x = \sum_{i=1}^{m} \theta_i x_i$ avec $\sum_i \theta_i = 1$ et $\theta_i \geq 0$, appartient à K.



Definition (Fonction Convexe)

Une fonction est dite convexe sur un convexe K si et seulement si $\forall x_0, x_1 \in K$,

$$f(\theta x_0 + (1 - \theta)x_1) \leq \theta f(x_0) + (1 - \theta)f(x_1).$$



Problèmes convexes

Definition (Problème Convexe)

Un problème d'optimisation est dit convexe lorsque f est convexe sur le domaine des points admissibles.

Soit K convexe fermé et f convexe sur K

$$\begin{aligned} &\min f(x) \\ &x \in K \end{aligned}$$

Proposition

Toute solution d'un problème convexe est une solution globale. L'ensemble des solutions est un ensemble convexe. Si f est strictement convexe, la solution est unique.

Gradient projeté

Soit K convexe fermé et f différentiable.

Algorithme du gradient projeté :

Pour k=0,1,...

$$x_0 \in K$$

$$x_{k+1} = \Pi_K(x_k - \rho g_k) \text{ avec } \Pi_K \text{ opérateur de projection orthogonale sur K}$$

Rappel : $\Pi_K(x) = \arg\min_{y \in K} \|x - y\|^2$

Le pas $\rho > 0$ est :

- soit fixe
  Converge si $\rho \leq 2\frac{\alpha}{M^2}$ avec α et M tels que :

$$(g(x) - g(y))^T(x - y) \geq \alpha\|x - y\|^2 \text{(coef. ellipticité)}$$
$$\|g(x) - g(y)\| \leq M\|x - y\| \text{(const. Lipschitz)}$$

  pour tout (x, y)
- soit choisi suivant le critère de convergence (cf méthodes de descente sans contrainte)

Remarque : On peut également utiliser un algorithme de type quasi-Newton avec projection.

Difficulté : Déterminer Π_K

Problèmes convexes : L'algorithme du Gradient projeté

Exemples :

- si $K = \{x : x \geq 0\}$ alors $\Pi_K(x) = \max(0, x)$
- si $K = \times_{i=1}^{n}[\alpha_i, \beta_i]$ $\Pi_K(x) = \min(\max(\alpha, x), \beta)$
- Autre exemple : projection sur un plan $P = \{x : Ax = b\}$

$$x^* = \Pi_P(x) = \arg\min_{y \in P} \|x - y\|^2$$

$$x^* = x - A^T(AA^T)^{-1}(Ax - b)$$

(exercice)

Si on ne connait pas Π_K ou si K n'est pas convexe, on peut passer par le problème dual.

Dualité

Hypothèse : $f \in C^1$, $c(x) \in C^1$

Soit le problème primal :

$$\text{Problème primal : } \min_x f(x)$$

$$\text{sous la contrainte } c(x) \geq 0$$

Le problème dual est défini par :

$$\text{Problème dual : } \max_\lambda \min_x \mathcal{L}(x, \lambda)$$

$$\text{sous la contrainte } \lambda \geq 0$$

Méthode Duale : Algorithme d'Uzawa

On utilise le fait que la solution du problème dual (x^*, λ^*) est un point selle de $L(x, \lambda)$:

$$L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*) \text{ pour tout } (x, \lambda)$$

et que x^* est également solution du problème primal.

Algorithme d'Uzawa :

$\lambda_0 \in C^+ = \{\lambda : \lambda \geq 0\}$ donné

Pour $k = 0, 1, 2, ...$

1. Pour λ_k fixé, on calcul : $x_k(\lambda_k) = \arg\min_x L(x, \lambda_k)$ (problème sans contrainte)
2. On applique un pas du gradient projeté pour maximiser $L(x_k(\lambda), \lambda)$ sous la contrainte $\lambda \geq 0$, soit :
   $\lambda_{k+1} = \Pi_{C^+}(\lambda_k - \rho c(x_k)) = \max(0, \lambda_k - \rho c(x_k))$
3. On retourne en 1 tant que le test d'arrêt n'est pas satisfait.

La fonction concave à maximiser au point 2 est

$$M(\lambda) = L(x_k(\lambda), \lambda) = \min_x L(x, \lambda)$$

Compte tenu que $\frac{\partial L}{\partial x} = 0$, on montre que son gradient vaut :

$$\nabla M(\lambda) = \frac{\partial L}{\partial \lambda} + \frac{\partial L}{\partial x}\frac{\partial x}{\partial \lambda} = -c(x)$$

Méthode Duale : l'Algorithme d'Uzawa

Converge si

1. $f, c \in C^1$
2. $\rho \leq \frac{2\alpha}{M^2}$ avec α et M tels que :

$$(g(x) - g(y))^T (x - y) \geq \alpha \|x - y\|^2 \text{(coef. ellipticité)}$$
$$\|g(x) - g(y)\| \leq M \|x - y\| \text{(const. Lipschitz)}$$

   pour tout (x, y)
3. Pour tout $\lambda \in C^+$, il existe un minimum unique x_λ à $L(x, \lambda)$,

Le point 3 est vérifié si les c_i sont concaves et si f est convexe.
Si $f \in C^2$, l'ellipticité correspond à

$$H(x) \geq \alpha Id, \text{ pour tout } x$$

Optimisation discrète : La programmation en nombre entier.

Problème du sac à dos : On considère un sac d'une capacité P que l'on cherche à remplir avec m produits de façon à maximiser l'utilité du chargement.

Si p_i désigne le poids de chaque objet et u_i son utilité, il nous faut donc résoudre

$$\max_x \sum_{i=1}^{m} u_i x_i \text{ sous les contraintes } \sum_{i=1}^{m} p_i x_i \leq P, \quad x_i \in \mathbb{N}.$$

Soit la formulation générale :

$$\begin{aligned} (P) \text{ Trouver } &: \min_x f(x) \\ & x_i \in \mathbb{Z} \\ & c(x) \geq 0 \end{aligned}$$

Optimisation discrète : La programmation en nombre entier.

Idée : On résout (P) sur $\mathbb{R}$. Soit $\hat{x}$ le minimum obtenu

- ou bien $\hat{x} \in \mathbb{Z}^n$ et on a fini
- ou bien $\hat{x} \notin \mathbb{Z}^n \Rightarrow$ il existe une composante $\hat{x}_i \notin \mathbb{Z}$. On crée alors deux sous problèmes (branches).

$$\begin{array}{cc} (P^-) : \min\limits_x f(x) & (P^+) : \min\limits_x f(x) \\ x \in \mathbb{R}, x_i \leq \mathbb{E}(\hat{x}_i) & x \in \mathbb{R}, x_i \geq \mathbb{E}(\hat{x}_i) + 1 \\ c(x) \geq 0 & c(x) \geq 0 \end{array}$$

Notation : $\mathbb{E}(x)$ désigne la partie entière de x

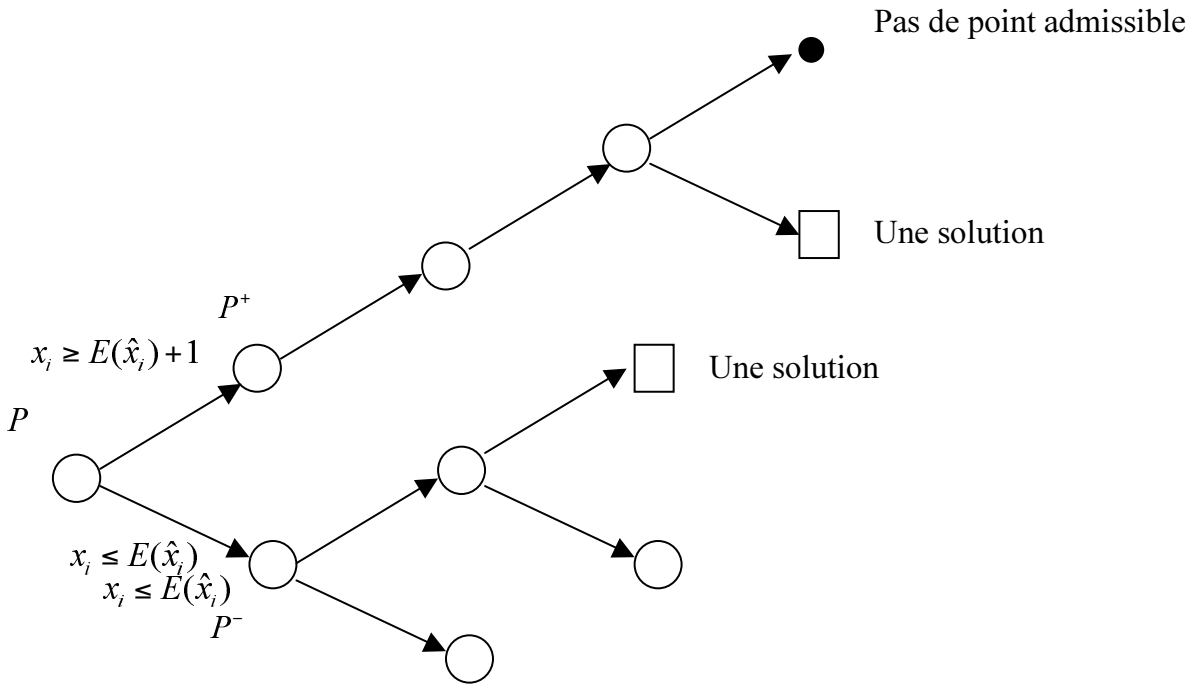
- Observation : Si on note x^- la solution de(P^-) et x^+ la solution de (P^+) alors

$$f(\hat{x}) \leq f(x^-)$$

$$f(\hat{x}) \leq f(x^+)$$

Optimisation discrète.

En itérant, on parvient à un arbre binaire de sous problèmes qui se termine soit par une solution partielle (locale) admissible □, soit par un problème sans point admissible •.



Idée : Il n'est pas nécessaire de développer tout l'arbre et de résoudre tous les sous problèmes car :

> si on trouve une solution locale □ avec un coût de f_i alors toute branche partant d'un noeud j tel que le coût associé $f_j > f_i$ donnera un résultat plus mauvais. Donc on n'explore pas ces branches.

Algorithme : "Branch and Bound"

On crée une pile FIFO (First In ,First Out) de sous problèmes (P) avec une borne inférieure L_P correspondant au coût du problème parent.
Pour les noeuds $\square$ qui ont été exploré, on note la meilleure solution $(\hat{x}, \hat{f})$.
On part de (P_0) : $\min_{x \in \mathbb{R}^n} f(x),\ c(x) \geq 0,\ L_{P_0} = -\infty$ et $\hat{f} = +\infty$.

1. Si la pile est vide, on arrête avec $(x^*, f^*) = (\hat{x}, \hat{f})$. Sinon on prend le problème (P_k) en haut de la pile (k représente le numéro du problème)
2. Si $L_k \geq \hat{f}$, on rejette le problème, retour en 1.
3. Sinon on essaye de résoudre le problème (P_k)
4. Si il n'y a pas de solution admissible, on rejette et on retourne en 1.
5. Sinon soit (x_k, f_k) la solution trouvée.
6. Si $f_k \geq \hat{f}$, on rejette le problème et retour en 1.
7. Sinon on sélectionne un indice i tel que $\mathbb{E}(x_{k_i}) < x_{k_i}$ et on crée deux sous problèmes (branches) $(P)_n$ et $(P)_{n+1}$ (n est le compteur de sous problèmes posés sur la pile) que l'on place en haut de la pile avec pour borne inférieure $L_n = L_{n+1} = f_k$ puis retour en 1.

Optimisation dynamique

Optimisation dynamique

- 1A Automatique :
  - Synthèse d'un retour d'état et d'un observateur par placement de pôles,
  - Synthèse de correcteurs simples (P, PI, avance de phase)
- L'objectif principale : Optimisation appliquée aux systèmes dynamiques
  $\rightarrow$ Synthèse d'une commande par optimisation d'un critère

- Problème de commande : Recherche d'une commande qui transfère un système dynamique :

$$\dot{x} = f(x, u) \qquad x_{k+1} = f(x_k, u_k)$$

d'un état initial x_0 à un état final x_T en minimisant un critère donné

$$J = \int_0^T L(x, u)dt \qquad J = \sum_{k=0}^{N} L(x_k, u_k)$$

- Outils :
  - Programmation non linéaire
  - Programmation dynamique
  - Principe du minimum

Les méthodes

- Les méthodes variationnelles : Ces méthodes reposent sur l'idée simple suivante :

  " Si une commande $\hat{u}$ conduisant à la valeur $\hat{J}$ est optimale, alors toute commande $u \neq \hat{u}$ donnera un résultat moins bon ".

- Si on peut évaluer l'effet d'une variation de la commande sur le critère : une commande $u = \hat{u} + \delta u$ conduit à un critère $J = \hat{J} + \delta J$, alors la condition $\delta J \geq 0$, $\forall \delta u$ permettra de caractériser $\hat{u}$.

  L'effet de δu sur J se fait par développement des solutions de l'équation d'évolution et du critère autour de la commande $\hat{u}$, candidate à être la commande optimale.

  On n'obtiendra donc en général que des conditions locales.

Les méthodes

- Les méthodes issues du principe de programmation dynamique que l'on peut énoncer grossièrement de la manière suivante :

  "à partir d'un point d'une trajectoire optimale, il faut minimiser ce qu'il reste du critère pour terminer la trajectoire".

- Exemples d'applications : Optimisation d'un critère quadratique, pour les systèmes linéaires (problème LQ).

Application de la PNL à la commande

Résoudre un problème de commande par la minimisation d'une fonction de plusieurs variables.

- L'optimisation paramétrique
- La commande d'un système à temps discret.

Optimisation paramétrique

L'optimisation paramétrique consiste à trouver les valeurs d'un ensemble de paramètres de manière à optimiser un critère.

- Problèmes d'identification : trouver les paramètres d'un modèle qui rendent compte au mieux du comportement d'un système.
- Problème de commande : trouver les meilleurs paramètres d'un correcteur.

Pour l'appliquer, il faut disposer du modèle du système à piloter, de la structure du correcteur que l'on va utiliser et du critère à optimiser.

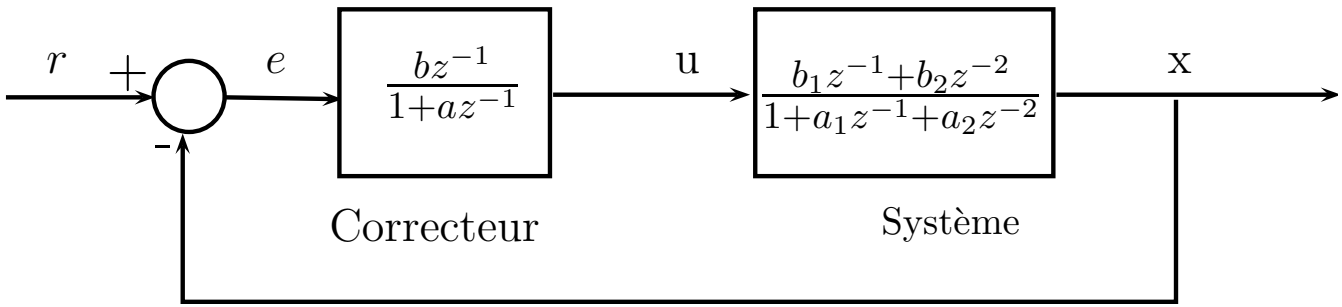
Avantages :

- Problèmes linéaires ou non linéaires
- Optimiser pour un type d'entrée (on n'est pas limité à l'échelon ou la sinusoïde).

Contre-parties :

- Déterminer le type de correcteur à utiliser (cours)
- Evaluation du gradient du critère à optimiser.

Exemple : Optimisation d'un correcteur



- Etant donné le procédé et une structure de correcteur,
- Etant donné une entrée choisie à l'avance,
- Etant donné un critère (fonction de coût), à minimiser,

quels sont les coefficients du correcteur qui vont minimiser le critère pour l'entrée donnée ?

Modèles mathématiques

- Equation du système :

$$x_{n+1} + a_1 x_n + a_2 x_{n-1} = b_1 u_{n-1} + b_2 u_{n-2} \tag{8}$$

- Equation du correcteur :

$$u_{n+1} + a u_n = b e_n \tag{9}$$

- l'erreur est :

$$e_n = r_n - x_n \tag{10}$$

- On optimise un critère de type somme :

$$J = \sum_{n=1}^{N} c(e_n, u_n) \tag{11}$$

où c est le coût élémentaire qui pénalise l'erreur et l'énergie fournie au système. N est l'horizon sur lequel on fait porter le coût ; il peut être infini.

- On choisira un critère quadratique pour des raisons de simplicité

$$J = \sum_{n=1}^{N} c(e_n, u_n) = \sum_{n=1}^{N} (e_n^2 + \lambda u_n^2) \tag{12}$$

  qui minimisera à la fois l'erreur et la commande.
- Le choix de λ impose la dynamique (plus ou moins rapide, plus ou moins amorti). En pratique, on fait des essais successifs.
- L'entrée r_n est donnée, elle devra être représentative des conditions réelles de fonctionnement du système, on prend souvent un échelon.

Minimisation du critère

- Le correcteur est caractérisé par son vecteur de paramètres : $p = [a, b]$. Les coefficients du correcteur seront optimaux lorsque le critère sera minimum.

$$\min_p \sum_{n=1}^{N} c(e_n(p), u_n(p))$$

- On minimisera le critère à l'aide d'un algorithme de PNL qui nécessitera le calcul du gradient ∇J du critère par rapport au vecteur paramètres.
- Le gradient est :

$$\nabla J = \sum_{i=1}^{N} \left(\frac{\partial c(e_n, u_n)}{\partial e_n} \frac{\partial e_n}{\partial p} + \frac{\partial c(e_n, u_n)}{\partial u_n} \frac{\partial u_n}{\partial p} \right) = \sum_{i=1}^{N} \left(2 e_n \frac{\partial e_n}{\partial p} + 2 \lambda u_n \frac{\partial u_n}{\partial p} \right)$$

- Les vecteurs $\frac{\partial e_n}{\partial p}$ et $\frac{\partial u_n}{\partial p}$ sont appelés : coefficients de sensibilité de e et de u par rapport aux coefficients du correcteur. Il faut les calculer.

- Par différenciation des équations dynamiques par rapport à p.
- Posons : $y_a = \frac{\partial x}{\partial a}; y_b = \frac{\partial x}{\partial b}; v_a = \frac{\partial u}{\partial a}; v_b = \frac{\partial u}{\partial b}$
- soit :

$$y_{an+1} + a_1 y_{an} + a_2 y_{an-1} = b_1 v_{an-1} + b_2 v_{an-2} \quad (13)$$

$$v_{an+1} + a v_{an} + u_n = -b y_{an} \quad (14)$$

et :

$$y_{bn+1} + a_1 y_{bn} + a_2 y_{bn-1} = b_1 v_{bn-1} + b_2 v_{bn-2} \quad (15)$$

$$v_{bn+1} + a v_{bn} = r_n - x_n - b y_{bn} \quad (16)$$

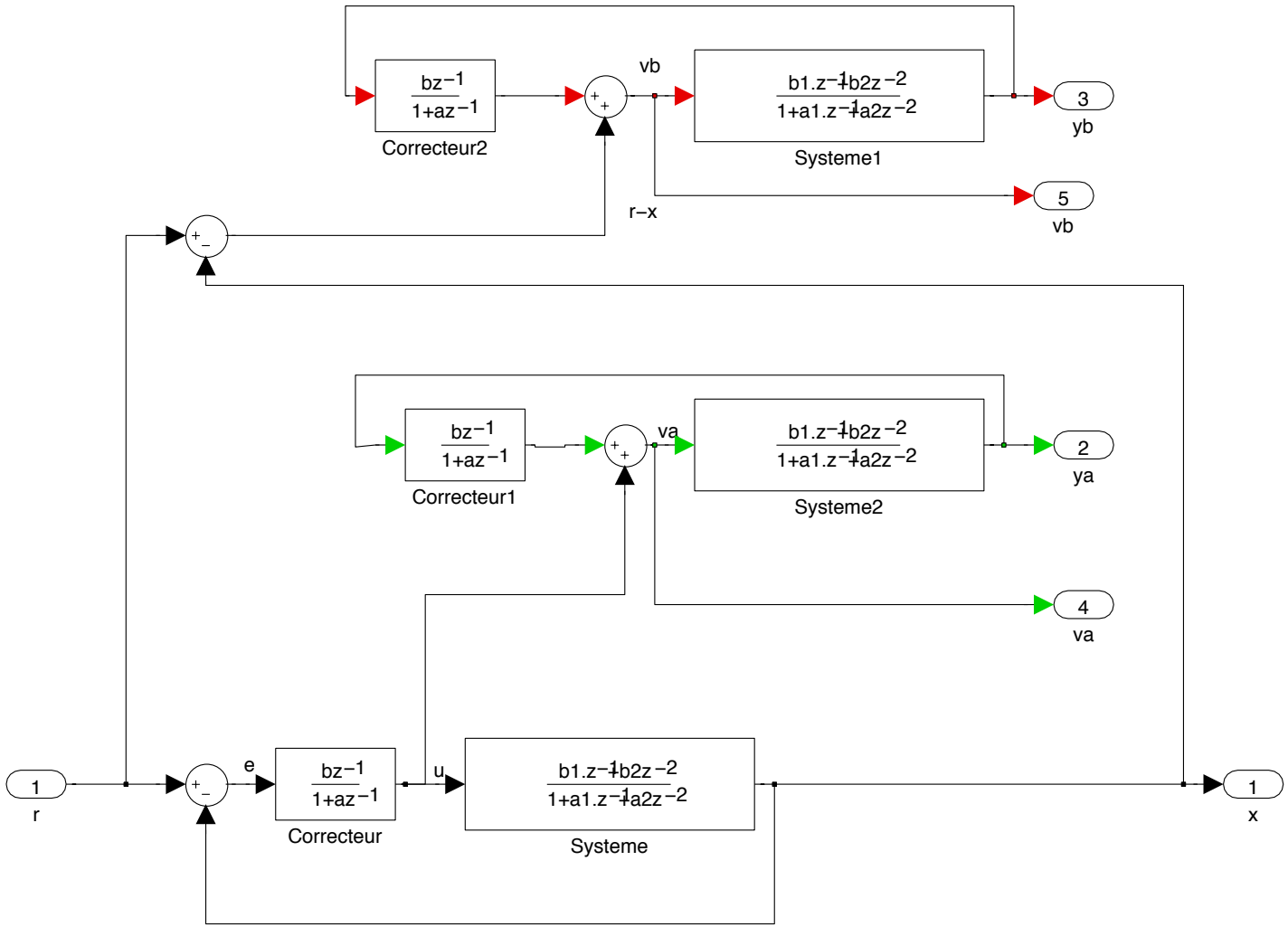
- Les conditions initiales sont toutes nulles.
- On résout en parallèle les équations du procédé et celles des fonctions de sensibilité
- Algorithme : Pour $k = 0, 1, ...$

$$(a, b)_k \mapsto (x, y, u, v)_k \mapsto (c, \nabla J)_k$$

$$(a, b)_{k+1} = (a, b)_k - \rho \nabla J$$

Fin lorsque $\|(a, b)_{k+1} - (a, b)_k\| \leq \epsilon$

Simulation des fonctions de sensibilité



Commande en temps discret

Les conditions d'optimalité de la PNL s'appliquent directement à la résolution des problèmes de commande optimale en temps discret.

Enoncé du problème :

On considère un système à temps discret dont l'évolution est représentée par une équation d'état :

$$x_{k+1} = f(x_k, u_k)$$

où $x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad k \in \mathbb{N}$

Il n'y a pas de contraintes sur u et x. Le critère à minimiser est de la forme :

$$J(x_0) = \sum_{k=0}^{N-1} c_k(x_k, u_k)$$

Nous supposerons que l'état initial x_0 est donné et que l'état final x_N doit satisfaire une contrainte :

$$x_N \in \mathcal{V}_f = \{x \in \mathbb{R}^n : \varphi(x) = 0\}$$

avec $\varphi = [\varphi_1 \ \varphi_2 \quad \varphi_p]^T$.

Solution

- Pour résoudre ce problème, l'idée est de considérer l'optimisation de J par rapport à l'ensemble des variables x et u.
- Les équations d'évolution et les conditions sur les états initial et final constituent les contraintes entre ces variables
- Soit $\mathcal{L}(x, u, \lambda, \mu)$ le Lagrangien :

$$\mathcal{L}(x, u, \lambda, \mu) = \sum_{k=0}^{N-1} c_k(x_k, u_k) + \sum_{k=0}^{N-1} \lambda_{k+1}^T (x_{k+1} - f(x_k, u_k)) + \mu^T \varphi(x_N) \quad (17)$$

Solution

- La stationnarité de $\mathcal{L}$ par rapport à toutes les variables donne le groupe d'équations suivant :

$$\frac{\partial \mathcal{L}}{\partial u_k} = 0 = \frac{\partial c_k(x_k, u_k)}{\partial u_k} - \frac{\partial f^T(x_k, u_k)}{\partial u_k}\lambda_{k+1}; k = 0, .., N-1 \quad (18)$$

$$\frac{\partial \mathcal{L}}{\partial x_k} = 0 = \frac{\partial c_k(x_k, u_k)}{\partial x_k} - \frac{\partial f^T(x_k, u_k)}{\partial x_k}\lambda_{k+1} + \lambda_k; k = 0, .., N-1 \quad (19)$$

$$\frac{\partial \mathcal{L}}{\partial x_N} = 0 = \lambda_N + \frac{\partial \varphi^T(x_N)}{\partial x_N}\mu \quad (20)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{k+1}} = 0 = x_{k+1} - f(x_k, u_k); k = 0, .., N-1 \quad (21)$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = 0 = \varphi(x_N) \quad (22)$$

- Pour résoudre ces équations, il faut pouvoir résoudre l'équation(18) en fonction de u_k : $u_k = \phi_k(x_k, \lambda_{k+1})$, expression que l'on reporte dans (19) et (21).
- On obtient ainsi un système autonome d'équations de récurrence en x et λ de dimension $2n$.

Solution

Les conditions aux limites sont données :

- en 0 par la donnée de x_0,
- en N par l'équation (20) qui indique que λ_N est combinaison linéaire des gradients des contraintes en x_N ou, si l'on préfère que λ_N est orthogonal à la variété $\mathcal{V}_f$ en x_N.(condition de transversalité).
- Remarquons que l'on aurait pu prendre comme hypothèse que $x_0 \in \mathcal{V}_0$ pour obtenir λ_0 orthogonal à la variété $\mathcal{V}_0$ en x_0.
- Lorsque l'état final est fixé, il n'y a pas de condition sur λ_N
- Lorsque l'état final est libre $\lambda_N = 0$.
  Ces conditions de transversalité nous indiquent que nous disposerons toujours de $2n$ conditions aux limites, mais malheureusement elles se répartiront en : n conditions en 0 et n conditions en N.

Exemple d'Algorithme

Problème avec coût terminal :

$$x_{k+1} = f(x_k, u_k), \quad x(0) = x_0$$

$$J = \Psi(x_N) + \sum_{k=0}^{N-1} c_k(x_k, u_k)$$

Algorithme :

- $u^0 = (u_0^0, u_1^0, \cdots, u_{N-1}^0)$ donné.
  Pour $i = 0, 1, ...$
  - (équation d'evolution) : Pour $k = 0, 1, ..., N-1$,

    $$x_{k+1}^i = f(x_k^i, u_k^i)$$

  - (condition finale) : $\lambda_N^i = -\frac{\partial \Psi}{\partial x}(x_N^i)$
  - (équation adjointe) : Pour $k = N-1, ..., 0$,

    $$\lambda_k^i = \frac{\partial f}{\partial x_k}^T (x_k^i, u_k^i)\lambda_{k+1}^i - \frac{\partial c_k(x_k^i, u_k^i)}{\partial x_k}$$

- (gradient sur équation de commande) : $u^{i+1} = u^i - \rho \frac{\partial \mathcal{L}(x^i, u^i, \lambda^i, \mu^i)}{\partial u}$
  (gradient ou autre chose)
- Fin si test d'arrêt satisfait : $\| \frac{\partial \mathcal{L}(x^i, u^i, \lambda^i, \mu^i)}{\partial u} \| < \epsilon$

Exemple : Commande LQ discrète

Position du problème : Soit le système linéaire :

$$x_{k+1} = Ax_k + Bu_k; x \in \mathbb{R}^n, u \in \mathbb{R}^p$$

et le critère quadratique à minimiser :

$$J(x_0) = \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2}x_N^T P_N x_N$$

x_0 donné $\quad x_N$ libre

avec $Q = Q^T \geq 0$, $R = R^T > 0$ $P_N = P_N^T \geq 0$

- Q peut s'écrire $Q = C^T C$; si Q est symétrique, elle est diagonalisable par une matrice orthogonale $Q = VDV^T$ avec $D = diag(\lambda_1, \lambda_2, ..., \lambda_r, 0, ...0)$ $\lambda_i > 0$

  Donc $C = diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}, ..., \sqrt{\lambda_r}, 0, ...0)V^T$ convient.

  Cette forme est souvent naturelle car on fait en général intervenir les sorties dans le critère : soit $y^T y$ avec $y = Cx$

Résolution par la programmation non-linéaire :

- Equations générales : Nous avons à trouver le minimum d'une fonction de $2N$ variables : $u_0, u_1, ..., u_{N-1}$ et $x_1, x_2, ..., x_N$ en présence des contraintes "égalité" formées par l'équation d'évolution.
- Le minimum sera obtenu en utilisant le Lagrangien $\mathcal{L}$ à l'aide des multiplicateurs de Lagrange.

$$\mathcal{L} = \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2}x_N^T P_N x_N + \sum_{k=0}^{N-1}\lambda_{k+1}^T(-x_{k+1} + Ax_k + Bu_k)$$

Equations générales :

En écrivant la stationnarité de $\mathcal{L}$ par rapport aux variables u, x et λ, nous obtenons trois groupes d'équations :

$$\frac{\partial \mathcal{L}}{\partial u_k} = 0 = Ru_k + B^T \lambda_{k+1}, \ k = 0, \cdots, N-1 \text{ équation de commande}$$
$$\frac{\partial \mathcal{L}}{\partial x_k} = 0 = Qx_k - \lambda_k + A^T \lambda_{k+1}, \ k = 0, \cdots, N-1 \text{ équation adjointe}$$
$$\frac{\partial \mathcal{L}}{\partial x_N} = 0 = P_N x_N - \lambda_N \text{ condition finale}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_{k+1}} = 0 = -x_{k+1} + Ax_k + Bu_k, \ k = 0, \cdots, N-1 \text{ équation d'évolution}$$

La commande optimale est donnée par :

$$\hat{u}_k = -R^{-1} B^T \lambda_{k+1}$$

en substituant, on trouve un système autonome de $2n$ équations de récurrence :

$$Qx_k - \lambda_k + A^T \lambda_{k+1} = 0$$
$$-x_{k+1} + Ax_k - BR^{-1} B^T \lambda_{k+1} = 0$$

Equations générales :

Soit encore :

$$\lambda_k = Qx_k + A^T \lambda_{k+1}$$
$$x_{k+1} = Ax_k - BR^{-1}B^T \lambda_{k+1}$$

- Pour le résoudre, il faut connaître $2n$ conditions aux limites. Nous connaissons x_0 soit n conditions. On a n relations à l'extrémité liant λ_N et x_N.

Solution : Equation de Riccati

La résolution progressive des équations de récurrence montre que λ peut s'écrire sous la forme :

$$\lambda_k = P_k x_k, \quad k = 0, \cdots, N \text{ à partir de } \lambda_N = P_N x_N$$

Cherchons alors directement une solution de cette forme.

$$\hat{u}_k = -R^{-1}B^T\lambda_{k+1} = -R^{-1}B^T P_{k+1}x_{k+1} = -R^{-1}B^T P_{k+1}(Ax_k + B\hat{u}_k)$$

soit :

$$\hat{u}_k = -[R + B^T P_{k+1}B]^{-1}B^T P_{k+1}Ax_k = -K_k x_k \tag{23}$$

La commande optimale est un retour d'état.

Equation de Riccati

En remplaçant λ_k et $\hat{u}_k$ par leur expression fonction de P_k dans l'équation adjointe $\lambda_k = Qx_k + A^T\lambda_{k+1}$, on trouve une équation de Riccati :

$$P_k x_k = A^T P_{k+1} A x_k - A^T P_{k+1} B[R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A x_k + Q x_k$$

La validité de cette équation quel que soit x_k est assurée si :

$$P_k = A^T P_{k+1} A - A^T P_{k+1} B[R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A + Q$$

$$P_k = A^T P_{k+1}(A - BK_k) + Q \text{ avec } K_k = [R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A$$

$$\text{ou } P_k = A^T P_{k+1} D_k + Q \text{ avec } D_k = A - BK_k$$

L'équation de Riccati se résout pas à pas à partir de la condition finale P_N, on en tire la commande optimale d'après

$$\hat{u}_k = -[R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A x_k = -K_k x_k.$$

L'equation d'évolution optimale est alors pour tout x_0 :

$$\begin{aligned} x_{k+1} &= D_k x_k \\ &= (A - BK_k) x_k \end{aligned}$$

Programmation dynamique

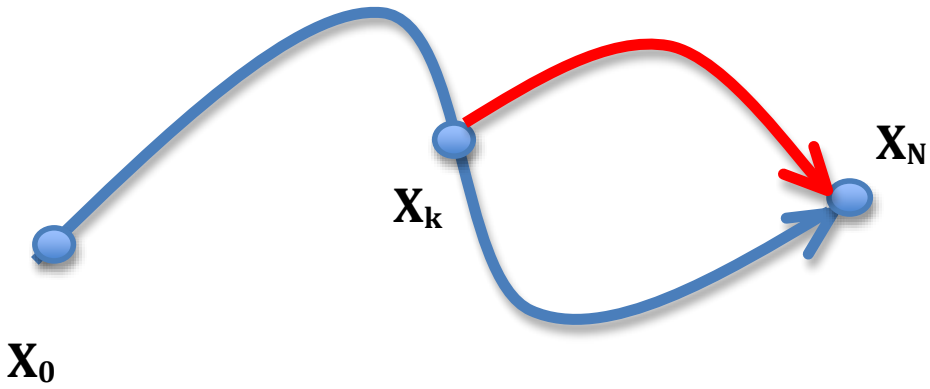
Programmation dynamique

- Le principe de Bellman (1920-1984) appliqué à des systèmes dynamiques
  Enoncé par Bellman dans les années 1950, ce principe débouche sur une gamme de stratégies de recherche d'optimum appelée "programmation dynamique".
- Recherche de chemin dans des graphes, dans les algorithmes de tri ou de classification, problème d'allocation de ressources
- Se formalise pour des systèmes dynamiques avec un critère d'optimisation de type additif.

Quelques exemples introductifs

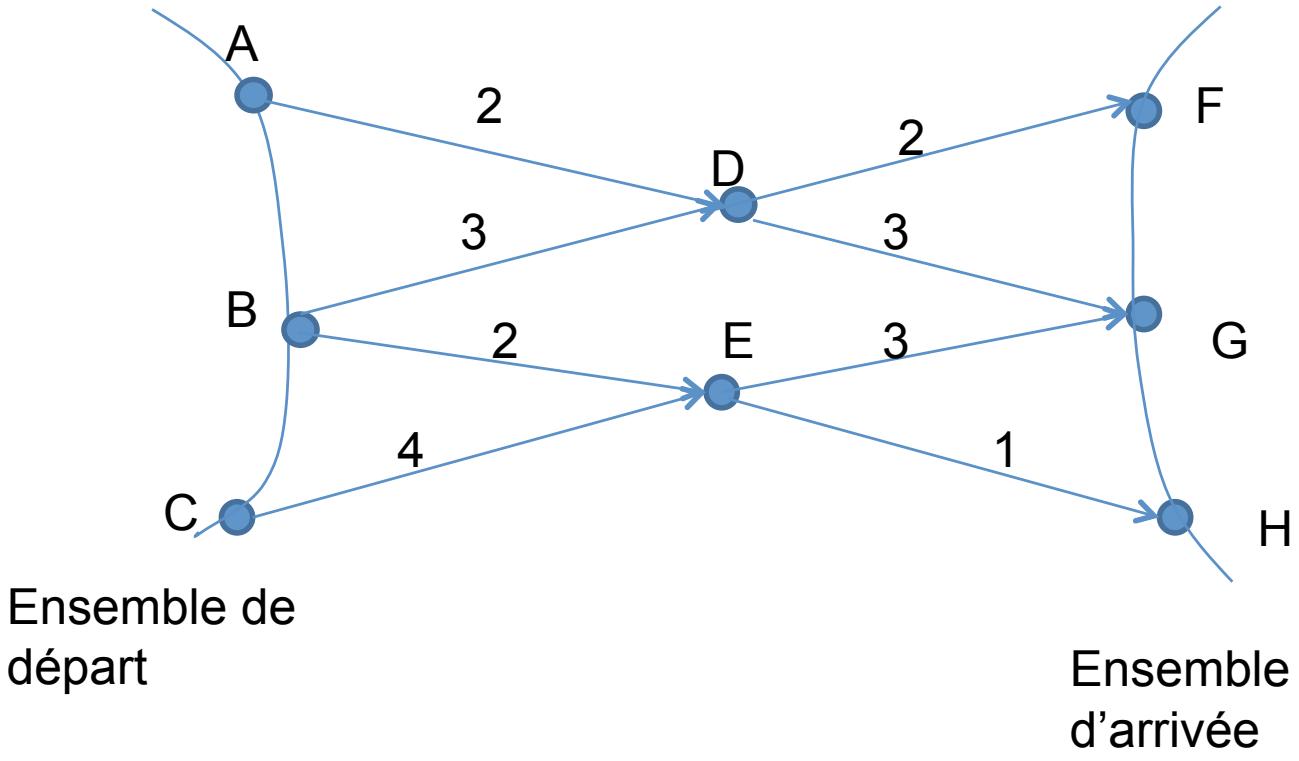
- Le principe d'optimalité

  "Un chemin optimal a la propriété que quelles que soient les conditions initiales et les commandes appliquées (choix effectués) sur une période initiale, la commande à appliquer sur la période restante doit être optimale pour le problème restant avec comme conditions initiales, l'état résultant de l'application des premières commandes".



- Deux exemples d'application du principe d'optimalité
  - Recherche du chemin minimum dans un graphe
  - Commande optimale d'un système à temps discret

Recherche du chemin minimum dans un graphe



- Objectif : On cherche le chemin de longueur minimum pour rejoindre l'ensemble de départ d={A,B,C} à l'ensemble d'arrivée a={F,G,H}

Recherche du chemin minimum dans un graphe

- Application du principe d'optimalité :
  - Si le chemin optimal passe par D, alors pour rejoindre {a} il faudra à partir de D prendre le chemin le plus court soit DF de longueur minimale $\hat{D}$=2.
  - Si le chemin optimal passe par E, alors pour rejoindre {a} il faudra à partir de E prendre le chemin le plus court soit EH de longueur minimale $\hat{E}$=1
  - Si le chemin optimal part de A, il passera par D et donc aura comme longueur $\hat{A}$=AD+$\hat{D}$=4
  - Si le chemin optimal part de B, il passera par D ou E et le meilleur sera donné par min(BD+$\hat{D}$,BE+$\hat{E}$) et donc aura comme longueur $\hat{B}$=3
  - Si le chemin optimal part de C, il passera par E et donc aura comme longueur $\hat{C}$=CE+E=5
  - Le chemin optimal sera obtenu en prenant min($\hat{A}$,$\hat{B}$,$\hat{C}$) soit BEH=3

Commande optimale d'un système à temps discret

Soit le système d'équation

$$x_{k+1} = ax_k + bu_n$$

On désire aller de l'état initial x_0 à l'état final $x_N = 0$ en minimisant le critère

$$J(x_0) = \sum_{n=0}^{N-1} \frac{1}{2} u_n^2$$

L'équation d'optimalité

En appliquant le principe d'optimalité, on déduit :

- Si à l'instant i, $0 < i < N$, l'état obtenu par les premières commandes $(u_0, u_1, ..., u_{i-1})$ est x_i, alors à partir de cet état il faut chercher la suite de commandes qui minimise

$$J_i(x_i) = \sum_{n=i}^{N-1} \frac{1}{2} u_n^2$$

et qui transfère le système à $x_N = 0$. Soit $\hat{J}(x_i)$ cette valeur optimale.

- En appliquant le même raisonnement pour résoudre ce nouveau problème et en prenant comme point intermédiaire l'instant $i + 1$, on déduit :

$$\hat{J}(x_i) = \min_{u_i}(\frac{1}{2} u_i^2 + \hat{J}(x_{i+1}))$$

$$x_{i+1} = ax_i + bu_i$$

- On voit qu'en appliquant cette récurrence depuis la fin, on pourra obtenir $\hat{J}(x_0)$ et la suite de commandes optimales à condition d'avoir mémorisé toutes les commandes optimales pour tous les états intermédiaires.

L'équation d'optimalité

Analyse : Si on cherche les points communs entre ces deux exemples, on peut conclure que dans les deux cas :

- Le critère est la somme des critères partiels ce qui a permis de conclure qu'il fallait minimiser le reste pour minimiser le tout,
- Pour résoudre le problème à partir du point intermédiaire sur la trajectoire, il n'était pas nécessaire de savoir comment ce point avait été atteint. Autrement dit, ce point représente un "état", on a donc affaire à un système dynamique.

Conclusion : Avec ces deux propriétés : critère additif (somme des coûts partiels) et système dynamique, le principe d'optimalité se démontre rigoureusement. C'est dans ce cadre que nous donnerons les résultats.

Définition système dynamique et d'un critère additif

Définition système dynamique Un système dynamique se caractérise par son espace d'état X, l'espace de temps T sur lequel il évolue (ensemble ordonné), son ensemble de fonctions de commande Ω, sa fonction de transition d'état φ :

$$\varphi = X \times T \times T \times \Omega \to X$$
$$x(t_2) = \varphi(x_1, t_1, t_2, u|_{[t_1,t_2]})$$

qui donne la valeur de l'état à l'instant t_2 lorsque l'état vaut x_1 à l'instant t_1 et que la commande u est appliquée sur l'intervalle $(t_1,\ t_2)$.
Axiome de concaténation : pour tout $t_2 \in [t_1, t_3]$,

$$\begin{aligned} x(t_3) &= \varphi(x_1, t_1, t_3, u|_{[t_1,t_3]}) \\ &= \varphi(\varphi(x_1, t_1, t_2, u|_{[t_1,t_2]}), t_2, t_3, u|_{[t_2,t_3]}) \end{aligned}$$

Axiome de consistence : $x_1 = \varphi(x_1, t_1, t_1, u|_{[t_1,t_1]})$

Critère additif
Lorsque le système évolue de t_1 à t_2 à partir de la condition initiale x_1 avec la commande u, on associe une fonction coût : $c(x_1, t_1, t_2, u|_{[t_1,t_2]})$.
Ce coût est dit additif si :

$$c(x_1, t_1, t_2, u|_{[t_1,t_2]}) = c(x_1, t_1, t_i, u|_{[t_1,t_i]}) + c(x(t_i), t_i, t_2, u|_{[t_i,t_2]}),\ \forall t_1 \leq t_i \leq t_2$$
$$avec\ x(t_i) = \varphi(x_1, t_1, t_i, u|_{[t_1,t_i]})$$

Problème de commande optimale

Problème (Commande optimale)

On considère un système dynamique et un coût additif, soit :

- $T_0 \subset T$ *un ensemble de temps initiaux,*
- $T_f \subset T$ *un ensemble de temps finals,*
- $X_0 \subset X$ *un ensemble d'états initiaux,*
- $X_f \subset X$ *un ensemble d'états finals,*

Déterminer $t_0 \in T_0,\ t_f \in T_f,\ x_0 \in X_0,\ x_f \in X_f, u \in \Omega$ *tel que* $x_f = \varphi(x_0, t_0, t_f, u)$ *de manière à ce que* $c(x_0, t_0, t_f, u|_{[t_0\ \ t_f]})$ *soit minimum.*

Principe de Bellman

Avec ces conditions, le principe de Bellman se démontre et se traduit par deux théorèmes.

Théorème

Si u^* est la commande optimale sur $[t_1, t_2]$ et x^* la trajectoire correspondante, alors les restrictions de u^*,$u^*_{[t_1,t_i]}$ et $u^*_{[t_i,t_2]}$ aux intervalles $[t_1, t_i]$ et $[t_i, t_2]$, sont les commandes optimales pour les deux problèmes :

1. *Problème $n^\circ 1$: partant des conditions initiales, atteindre $x^*(t_i)$ en minimisant le critère $c(x_1, t_1, t_i, u_{[t_1,t_i]})$,*
2. *Problème $n^\circ 2$: partant de $x^*(t_i)$, atteindre les conditions finales en minimisant le critère $c(x^*(t_i), t_i, t_2,_{[t_i,t_2]})$.*

Equation d'optimalité

De ce théorème, en envisageant toutes les valeurs possibles x_i pour l'état à l'instant t_i et en cherchant le minimum, on en déduit le deuxième théorème.

Théorème

Etant donnés trois instants $t_1 \leq t_i \leq t_2$, et les conditions aux limites (x_1, t_1) et (x_2, t_2), on a l'équation d'optimalité suivante :

$$\min_{u_{[t_1,t_2]}} c(x_1, t_1, t_2, u_{[t_1,t_2]}) = \min_{u_{[t_1,t_i]}} \left(c(x_1, t_1, t_i, u_{[t_1,t_i]}) + \min_{u_{[t_i,t_2]}} c(x_i, t_i, t_2, u_{[t_i,t_2]}) \right)$$

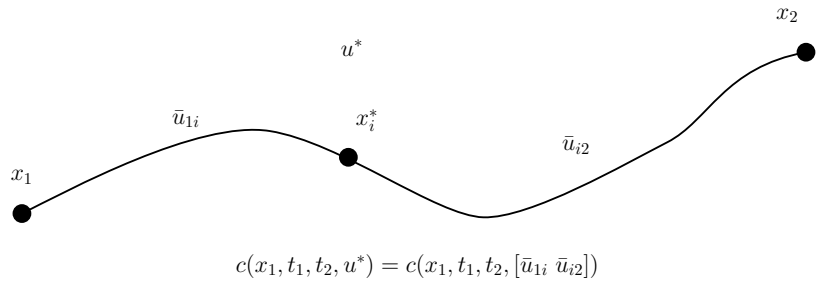
$$\text{avec } x_i = \varphi(x_1, t_1, t_i, u_{[t_1,t_i]})$$

$$x_2 = \varphi(x_i, t_i, t_2, u_{[t_i,t_2]})$$

Autrement dit :

Pour minimiser le coût global de t_1 à t_2 , il faut minimiser la somme formée du coût de transfert de t_1 à t_i et du coût minimal pour aller de t_i à t_2.

démonstration Théorème 1



Soit u^* est la commande optimale sur $[t_1, t_2]$, et $x^*(t) = \varphi(x_1, t_1, t, u^*_{[t_1,t]})$, la trajectoire optimale.
Posons

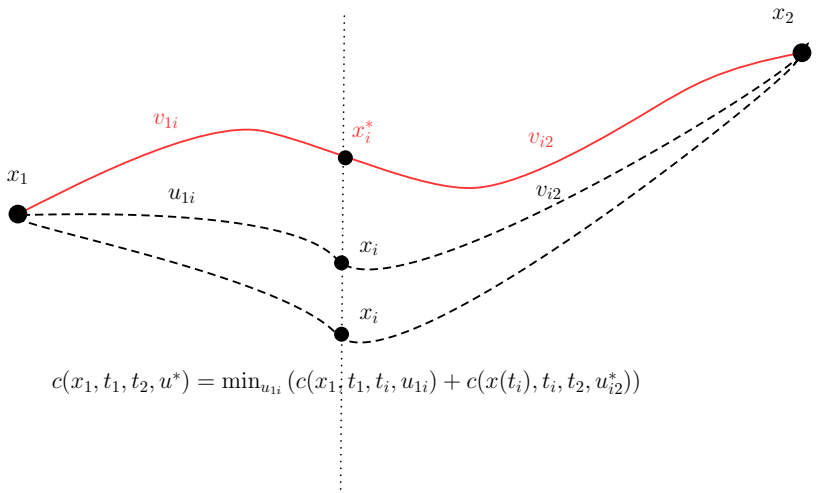
$$x_i^* = x^*(t_i).$$

- Soit $\bar{u}_{[t_1\ \ t_i]}$ une commande qui transfère le système de (t_1, x_1) à (t_i, x_i^*) en minimisant $c(x_1, t_1, t_i, u)$
- Soit $\bar{u}_{[t_i\ \ t_2]}$ une commande qui transfère le système de (t_i, x_i^*) à (t_2, x_2^*) en minimisant $c(x_i^*, t_i, t_2, u)$
- La commande $\bar{u}$, concaténation de $\bar{u}_{[t_1\ \ t_i]}$ et $\bar{u}_{[t_i\ \ t_2]}$ transfère le système de (t_1, x_1) à $(t_2,\ x_2^*)$ en passant par x_i^*.
- Par additivité des coûts, on a :

$$\begin{aligned} c(x_1, t_1, t_2, \bar{u}) &= c(x_1, t_1, t_i, \bar{u}_{1i}) + c(x_i^*, t_i, t_2, \bar{u}_{i2}) \\ &\leq c(x_1, t_1, t_i, u^*|_{(t_1,t_i)}) + c(x_i^*, t_i, t_2, u^*|_{(t_i,t_2)}) \\ &= c(x_1, t_1, t_2, u^*) \end{aligned}$$

l'inégalité stricte ne peut donc pas avoir lieu.

Démonstration Théorème 2



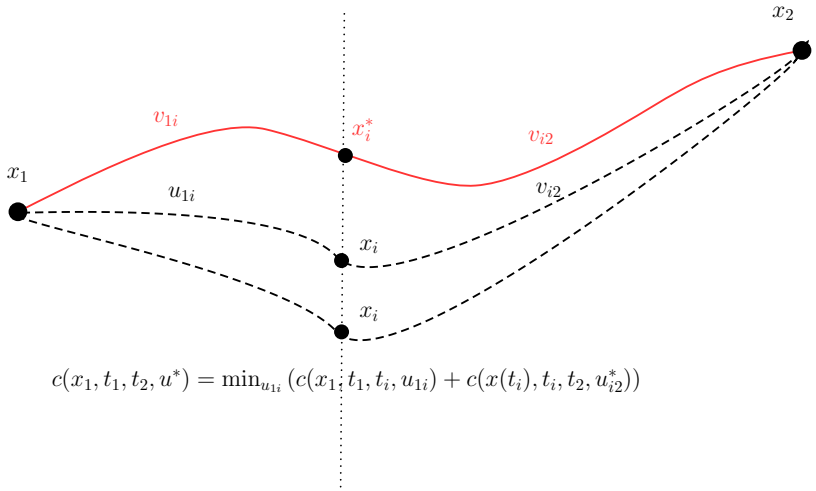
- On considère l'ensemble V des commandes $v(x_i)$ pour aller de (t_1, x_1) à (t_2, x_2^*) en passant par (t_i, x_i) avec $x_i \in \{\text{états possibles en } t_i\}$, et telles que $v_{[t_1,t_i]}$ et $v_{[t_i,t_2]}$, restrictions de v aux intervalles $[t_1, t_i]$ et $[t_i, t_2]$ soient optimales pour chaque sous problème.
- $x_i^* \in \{$ états possibles en $t_i\}$, et $u^* \in V(x_i^*)$ d'après le théorème 1.

- Par définition de $v(x_i)$:

$$c(x_1, t_1, t_2, v(x_i)) = \min_{u_{[t_1,t_i]} \,:\, x(t_i)=x_i} c(x_1, t_1, t_i, u_{[t_1,t_i]}) + \min_{u_{[t_i,t_2]} \,:x(t_2)=x_2} c(x_i, t_i, t_2, u_{[t_i,t_2]})$$

$$c(x_1, t_1, t_2, u^*) = \min_{x_i} c(x_1, t_1, t_2, v(x_i))$$

Démonstration Théorème 2



$$c(x_1, t_1, t_2, u^*) = \min_{x_i} c(x_1, t_1, t_2, v(x_i))$$

$$= \min_{x_i} \left(\min_{u_{1i} \ : \ x(t_i)=x_i} c(x_1, t_1, t_i, u_{1i}) + \min_{u_{i2} \ : \ x(t_2)=x_2} c(x_i, t_i, t_2, u_{i2}) \right)$$

$$= \min_{x_i} \left(\min_{u_{1i} \ : \ x(t_i)=x_i} \left(c(x_1, t_1, t_i, u_{1i}) + \min_{u_{i2} \ : \ x(t_2)=x_2} c(x_i, t_i, t_2, u_{i2}) \right) \right)$$

$$= \min_{x_i} \left(\min_{u_{1i} \ : \ x(t_i)=x_i} \left(c(x_1, t_1, t_i, u_{1i}) + \min_{u_{i2} \ : \ x(t_2)=x_2} c(x(t_i), t_i, t_2, u_{i2}) \right) \right)$$

en relachant la contrainte $x(t_i) = x_i$

$$\geq \min_{u_{1i}} \left(c(x_1, t_1, t_i, u_{1i}) + \min_{u_{i2} \ : \ x(t_2)=x_2} c(x(t_i), t_i, t_2, u_{i2}) \right)$$

Synthèse

Pour résoudre un problème en utilisant la programmation dynamique, il faut :

1. Mettre le problème dans la formulation "système dynamique, critère additif". Il faut donc :
   - Identifier un processus dynamique de décision (temps, commande, état)
   - Trouver l'équation d'évolution
   - Exprimer le critère
2. Initialiser la récurrence en partant de la fin
3. Résoudre l'équation d'optimalité (Théorème 2) en temps rétrograde

Exemples : Systèmes à temps discret

- On considère un système à temps discret dont l'évolution est représentée par une équation d'état :

$$x_{k+1} = f(x_k, u_k),\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m, k \in \mathbb{N}$$

- Il n'y a pas de contraintes sur u et x. Le critère à minimiser est de la forme :

$$J(x_0, u_0, \cdots, u_{N-1}) = \sum_{k=0}^{N-1} c_k(x_k, u_k) + \Psi(x_N)$$

- Nous supposerons que l'état final x_N est libre, l'état initial x_0 est donné.
- Les hypothèses pour appliquer la programmation dynamique sont trivialement vérifiées.

Algorithme

- A l'instant intermédiaire p, notons par :

$$\hat{J}_p(x_p) := \min_{u_p, u_{p+1}, \ldots, u_{N-1}} \sum_{k=p}^{N-1} c_k(x_k, u_k) + \Psi(x_N)$$

  le critère optimal pour aller à la fin ($p \to N$) depuis l'état x_p.

- Appliquons l'équation d'optimalité sur les instants p, $p+1$ et N :

$$\text{Pour tout } x_p, \ \hat{J}_p(x_p) = \min_{u_p} \left(c_p(x_p, u_p) + \hat{J}_{p+1}(x_{p+1}) \right)$$

$$\text{avec } x_{p+1} = f(x_p, u_p)$$

- Initialisation (p=N) :

$$\hat{J}_N(x_N) = \Psi(x_N)$$

- La résolution de cet algorithme donne la commande optimale en boucle fermée $\hat{u}_p(x_p)$. Les équations sont résolues en sens rétrograde de $p = N-1$ jusque $p = 0$ avec la condition initiale.

Exemples : Plan d'investissement

On se propose de faire une campagne de publicité dans 4 régions. On possède un budget de 5 unités. On investit un certain nombre d'unités dans chaque région le rendement étant donné dans le tableau suivant :

	I	II	III	IV
0	0	0	0	0
1	0,28	0,25	0,15	0,20
2	0,45	0,41	0,25	0,35
3	0,65	0,55	0,40	0,42
4	0,78	0,65	0,50	0,48
5	0,90	0,75	0,62	0,53

Déterminer la stratégie optimale d'investissement.

Formalisation :

- Temps : k=1,2,3,4 (+5) Initialisation
- Etats : x_k nombre d'unités de Budget restant à investir à l'étape k
- Commandes : u_k nombre d'unités de Budget investi dans la région k

Exercice : Ecrire l'algorithme complet
(Eq. d'état + Eq. d'optimalité + Initialisation)

Chemin dans un graphe

- Lorsque l'espace d'état contient un nombre fini d'éléments, la dynamique peut être représentée par un graphe.
- Chaque sommet i est un état, les arcs donnent la fonction de transition d'état. Les arcs sont valués par le coût de leur franchissement : c_{ij} est le coût pour franchir l'arc (i, j).
- La commande est le chemin choisi pour aller du sommet origine à l'extrémité.
- Pour introduire le coût terminal, il suffit d'introduire un état fictif t et le coût c_{it} lorsque i est l'état d'arrivée.

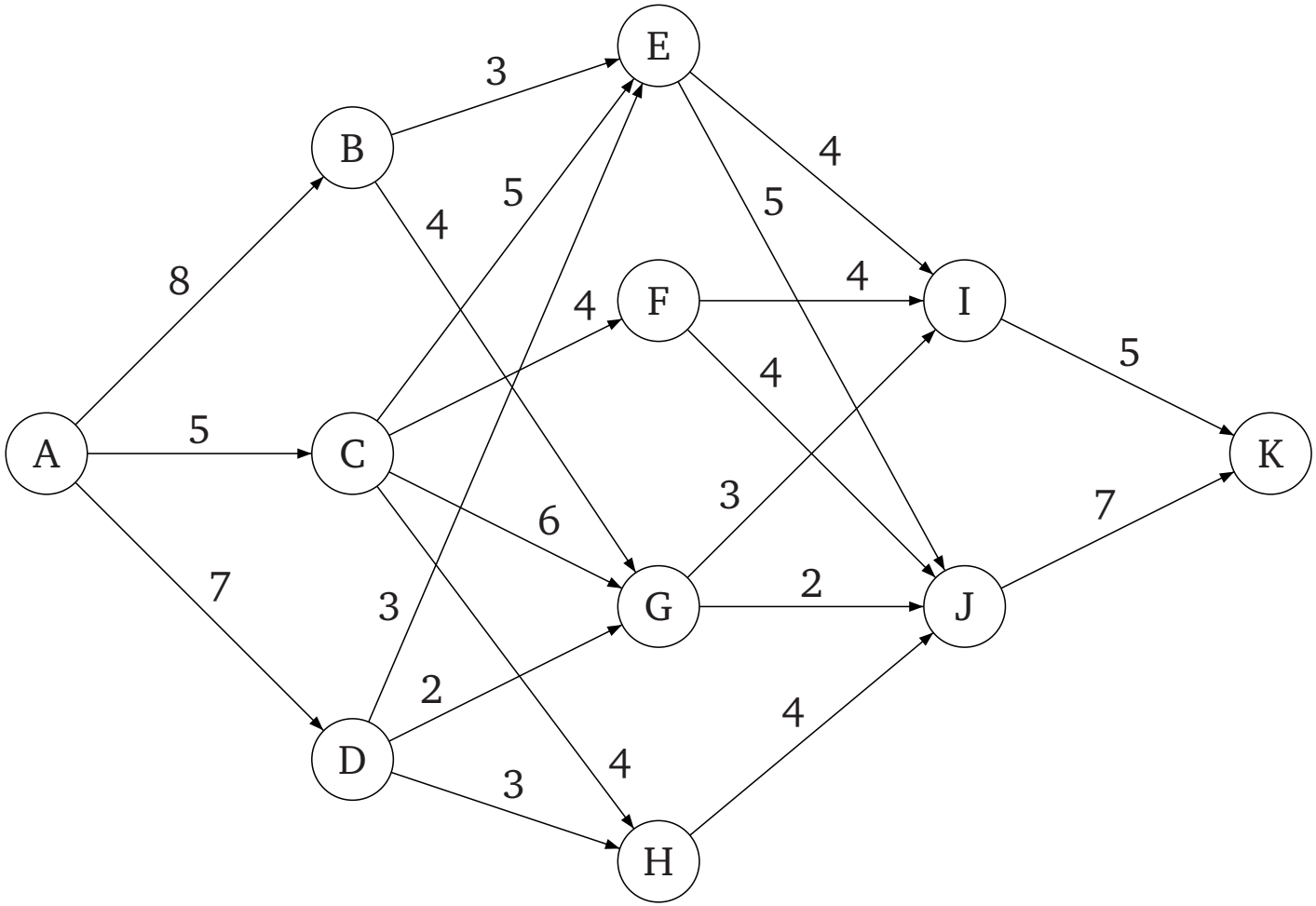
Algorithme

- On appellera S_k l'ensemble des états accessibles à l'étape k.
- L'algorithme de recherche du chemin minimum est alors :
  1. $\hat{J}_N(i) = c_{it},\ \forall i \in S_N$ (initialisation)
  2. $\hat{J}_k(i) = \min_{j \in S_{k+1}} (c_{ij} + \hat{J}_{k+1}(j))\ \forall i \in S_k$ (équation d'optimalité)
  3. $\hat{J}_0(i)$ est le chemin minimum pour aller de i à l'espace d'arrivée.(fin)

Remarque

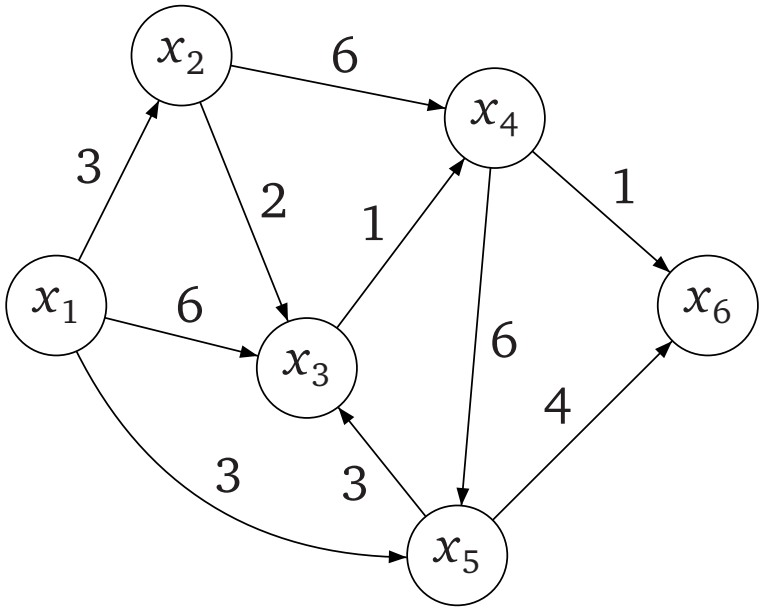
On suppose qu'il n'existe pas de cycle à coût négatif, le nombre maximum de sommets par lesquels passe le chemin minimum est constitué au plus de N sommets. En posant $c_{ii} = 0$, *on pourra toujours considérer qu'il y a N étapes.*

Exemple 1 :



Déterminer le plus court chemin joignant A à K.

Exemple 2 :



Déterminer le plus court chemin joignant x_1 à x_6.

Cas stochastique

La programmation dynamique peut aussi se formaliser dans le cas stochastique.
Enoncé du problème : Soit le système

$$x_{k+1} = f(x_k, u_k, w_k); x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ k \in \mathbb{N}$$

x_0 donné.
Les w_k sont des variables aléatoires indépendantes (leur loi de probabilité peut dépendre de x_k et u_k).

$$P(w_k \mid x_k, u_k)$$

Les commandes sont contraintes $u_k \in U_k$.
Le coût est additif :

$$J = \underset{w_0, w_1, .., w_{N-1}}{\mathbb{E}} \{\sum_{k=0}^{N-1} c_k(x_k, u_k, w_k) + \Psi(x_N)\}$$

Algorithme

Hypothèses : La loi de probabilité sur les coûts induite par les v.a. w_k admet une moyenne finie quelle que soit la loi de commande $u_k = u_k(x_k)$.

Le problème consiste à trouver la loi de commande optimale u^* qui minimisera J soit $J^*(x_0)$.

Algorithme

Pour tout x_N, $J_N^*(x_N) = \Psi(x_N)$(initialisation)

Pour tout x_k,(équation d'optimalité)

$$J_k^*(x_k) = \min_{u_k \in U_k} \mathop{\mathbb{E}}_{w_k}(c_k(x_k, u_k, w_k) + J_{k+1}^*(f(x_k, u_k, w_k))$$

$$J^*(x_0) = J_0^*(x_0)\text{(fin)}$$

Retour sur la commande LQ en temps discret : Résolution par la programmation dynamique

- Posons $J_p(x_p) = \frac{1}{2}\sum_{k=p}^{N-1}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2}x_N^T P_N x_N$
- Le problème est de minimiser $J_0(x_0)$ avec une dynamique déterminée par :

$$x_{k+1} = Ax_k + Bu_k, \;\; k = 0, 1, 2, \cdots \quad x_0 \text{ donné et } u_k \in \mathbb{R}^m$$

Posons $\hat{J}_p(x_p) = \min_{u_p, u_{p+1}, \ldots, u_{N-1}} J_p(x_p)$ où x_p est l'état à l'instant p

- L'algorithme de programmation dynamique donne :

$$\hat{J}_N(x_N) = \frac{1}{2}x_N^T P_N x_N$$

$$\hat{J}_k(x_k) = \min_{u_k \in \mathbb{R}^m}\{\frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + \hat{J}_{k+1}(Ax_k + Bu_k)\}$$

$$\hat{J}(x_0) = \hat{J}_0(x_0)$$

Commande LQ en temps discret : Résolution de l'équation d'optimalité

- En appliquant l'équation de récurrence, le calcul de $\hat{J}_{N-1}(x_{N-1})$ est obtenu en calculant explicitement le minimum

  Possible dans le cas sans contrainte en annulant le gradient. On obtient une équation linéaire en u puisqu'on dérive une forme quadratique.

- Les résultats sont les suivants :

$$\frac{\partial J_{N-1}(x_{N-1}, u_{N-1})}{\partial u_{N-1}} = 0$$

$$\text{soit } Ru_{N-1} + B^T P_N(Ax_{N-1} + Bu_{N-1}) = 0 \text{ (\textit{exercice})}$$

$$R + B^T P_N B > 0 \text{ \textit{car} } R > 0$$

d'où le résultat :

$$\hat{u}_{N-1} = -[R + B^T P_N B]^{-1} B^T P_N A x_{N-1} = -K_{N-1} x_{N-1}$$

$$\hat{J}_{N-1}(x_{N-1}) = \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1}$$

$$\text{avec } P_{N-1} = A^T P_N A + Q - A^T P_N B[R + B^T P_N B]^{-1} B^T P_N A \text{ (\textit{exercice})}$$

Commande LQ en temps discret : Résolution de l'équation d'optimalité

Prenons comme hypothèse de récurrence :

$$\hat{J}_k(x_k) = \frac{1}{2} x_k^T P_k x_k$$

En appliquant la même démarche que précédemment, nous obtiendrons de manière évidente les mêmes résultats en substituant k à N, soit :

$$\begin{aligned}
\hat{u}_k &= -[R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A x_k = -K_k x_k \\
\hat{J}_k(x_k) &= \frac{1}{2} x_k^T P_k x_k \\
P_k &= A^T P_{k+1} A + Q - A^T P_{k+1} B [R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A \quad (*)
\end{aligned}$$

On retrouve le résultat obtenu par PNL (Ouf !).

Commande LQ en temps discret : Equation de Riccati

L'équation de récurrence est :

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B [R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A$$

avec P_N donné

$$\hat{u}_k = -K_k x_k \text{ avec } K_k = [R + B^T P_{k+1} B]^{-1} B^T P_{k+1} A$$
$$x_{k+1} = (A - BK_k) x_k$$

Le critère minimisé est :

$$J_0 = \frac{1}{2} x_N^T P_N x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$

la valeur du critère optimal est :

$$\hat{J}_0(x_0) = \frac{1}{2} x_0^T P_0 x_0$$

Remarque

Nous avons une structure de retour d'état (non constant), donc une boucle fermée. La matrice d'état, à paramètres variables est $D_k = A - BK_k$

Commande LQ en temps discret : Comportement asymptotique

Quand les matrices A, B, Q et R sont constantes et quand N tend vers l'infini, P_0 tend vers une solution stationnaire qui satisfait l'équation algébrique de Riccati :

$$P = A^T PA + Q - A^T PB[R + B^T PB]^{-1} B^T PA$$

Théorème

Soit le système (A, B, C), et le critère

$$J = \frac{1}{2} \sum_0^\infty (x_k^T Q x_k + u_k^T R u_k), \ Q = C^T C \geq 0, \ R = R^T > 0$$

Si (A, B) est commandable et (A, C) est observable alors :

1. *P est l'unique matrice $P = P^T > 0$ solution de l'équation algébrique de Riccati*
2. *Le critère optimal est*
$$J^*(x_0) = \frac{1}{2} x_0^T P x_0$$
3. *La commande optimale est un retour d'état de gain K :*
$$\hat{u}_k = -Kx_k \text{ avec } K = [R + B^T PB]^{-1} B^T PA$$
4. *Le système optimal vérifie l'équation $x_{k+1} = Dx_k$, avec $D = A - BK$*
5. *Ce système est asymptotiquement stable.*

Convergence de ER vers une solution stationnaire

Passage à la limite (Cas sans critère terminal $P_N = 0$)

- L'ER est :

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B [R + B^T P_{k+1} B]^{-1} B^T P_k A$$

avec $P_N = 0$

- On minimise $\frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$, le résultat optimal en N étapes est $\hat{J_0}^N(x_0) = \frac{1}{2} x_0^T P_0^N x_0$.
- Montrons que le critère optimal en N étapes $\hat{J_0}^N(x_0)$ est inférieur au critère optimal en $N+1$ étapes $\hat{J_0}^{N+1}(x_0)$pour tout N et x_0.

Soit $\hat{u}_{0,N+1}, \hat{u}_{1,N+1}, ..\hat{u}_{N,N+1}$ la commande optimale en $N+1$ étapes,

$$\begin{aligned}
\frac{1}{2} x_0^T P_0^{N+1} x_0 = \hat{J_0}^{N+1}(x_0) &= \frac{1}{2} \sum_{k=0}^{N} (\hat{x}_{k,N+1}^T Q \hat{x}_{k,N+1} + \hat{u}_{k,N+1}^T R \hat{u}_{k,N+1}) \\
&\geq \frac{1}{2} \sum_{k=0}^{N-1} (\hat{x}_{k,N+1}^T Q \hat{x}_{k,N+1} + \hat{u}_{k,N+1}^T R \hat{u}_{k,N+1}) \\
&\geq \min_{u_0,\ldots,u_{N-1}} \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \\
&= \hat{J_0}^N(x_0) = \frac{1}{2} x_0^T P_0^N x_0
\end{aligned}$$

Démonstration : solution stationnaire

- La suite $N \mapsto \hat{J}_0^N(x_0) = \frac{1}{2} x_0^T P_0^N x_0$ est donc non décroissante, $\forall x_0$.
- La quantité $\hat{J}_0^N(x_0) = \frac{1}{2} x_0^T P_0^N x_0$ est bornée supérieurement par le coût obtenu en appliquant les commandes u_k qui ramènent le système à zéro en n_0 étapes (existe car le système est commandable) et $u_k = 0$, $k \geq n_0$ ensuite, donc cette suite $N \mapsto \hat{J}_0^N(x_0)$ converge $\forall x_0$.
- Ceci entraîne la convergence de tous les termes de la matrice P_0^N vers une matrice constante P symétrique non négative.

$$P_0^N \to P, \quad N \to +\infty$$

- En passant à la limite dans l'équation de Riccati $P_k = f(P_{k+1})$, on obtient l'équation de Riccati algébrique $P = f(P)$ (point fixe).

Stabilité de la boucle fermée

Montrons que la fonction $x \to V(x) = x^T P x$ est une fonction de Lyapounov du système.

- Calculons $\Delta V_k = V(x_{k+1}) - V(x_k)$.
- L'équation de Riccati peut aussi s'écrire :

$$\begin{aligned} P &= D^T P D + Q + K^T R K \\ \text{avec } D &= A - BK \\ \text{et } K &= [R + B^T P B]^{-1} B^T P A \end{aligned}$$

- D'où :

$$\Delta V_k = x_{k+1}^T P x_{k+1} - x_k^T P x_k = x_k^T (D^T P D - P) x_k = -x_k^T (Q + K^T R K) x_k \leq 0$$

- En général, on n'a pas $\Delta V_k < 0$, on ne peut donc pas conclure directement que $x_k \to 0$.

Stabilité de la boucle fermée

- En itérant l'équation ΔV_k, on obtient :

$$x_{k+1}^T P x_{k+1} - x_0^T P x_0 = -\sum_{i=0}^{k} x_i^T (Q + K^T R K) x_i$$

Soit encore $x_{k+1}^T P x_{k+1} + \sum_{i=0}^{k} x_i^T (Q + K^T R K) x_i = x_0^T P x_0$

- on sait que $x_{k+1}^T P x_{k+1} \geq 0$ donc la série $\sum_{i=0}^{k} x_i^T (Q + K^T R K) x_i$ converge et donc $x_k^T (Q + K^T R K) x_k \to 0$ quand $k \to +\infty$.

$$\text{donc } C x_k \to 0 \text{ et } K x_k \to 0 \text{ quand } k \to +\infty$$

Stabilité de D=A-B K

L'hypothèse d'observabilité de (A, C) va nous permettre de conclure que $x_k \to 0$.
En effet :

$x_k = Ix_k$

$x_{k+1} = Ax_k - BKx_k$

$x_{k+2} = A^2 x_k - ABKx_k - BKx_{k+1}$

$\vdots$

$x_{k+n-1} = A^{n-1} x_k - .....$

En multipliant toutes les lignes à gauche par C on déduit :

$Cx_k = Cx_k$

$Cx_{k+1} + CBKx_k = CAx_k$

$Cx_{k+2} + CABKx_k + CBKx_{k+1} = CA^2 x_k$

$\vdots$

$Cx_{k+n-1} + C + ... = CA^{n-1} x_k$

Les termes à gauche du signe $=$ tendent vers zéro, donc $\mathcal{O}(A, C), \quad x_k \to 0$ donc x_k tend vers zéro quand $k \to \infty$.

P défini positif

- Il est évident que $P \geq 0$ puisque le critère ne peut pas être négatif.
- Supposons qu'il existe $x_0 \neq 0$ tel que $x_0^T P x_0 = 0$ alors comme

$$x_0^T P x_0 = \sum_{i=0}^{+\infty} x_i^T (Q + K^T R K) x_i$$

on a :

$$x_k^T (Q + K^T R K) x_k = 0 \ \forall k,$$

donc $C x_k = 0 \ \forall k$, d'où $x_k = 0$ et $x_0 = 0$ par l'observabilité de (A, C), ce qui contpurpleit l'hypothèse.

Principe du minimum (Pontriaguine)

Principe du minimum (Pontriaguine)

Formalisation du problème :

- Soit l'équation différentielle (temps invariant) :

$$\dot{x} = f(x,u), \qquad x(t_0) = x_0 \qquad x(t_f) = x_f$$

avec $x(t) \in \mathbb{R}^n$, $u(t) \in U \subset \mathbb{R}^m$, $x_0 \in C_0$ et $x_f \in C_f$.

- Problème de commande optimale : Déterminer $u(.)$, x_0, t_f, x_f tels que la fonction critère :

$$J = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t),u(t))dt$$

soit minimisée sous les contraintes

$$x_0 \in C_0 = \{x : c_{0,i}(x) = 0,\ \ c_{0,i} : \mathbb{R}^n \to \mathbb{R}, i = 1,...,p_0\},$$
$$x_f \in C_f = \{x : c_{f,i}(x) = 0,\ \ c_{f,i} : \mathbb{R}^n \to \mathbb{R}, i = 1,...,p_f,\}$$

- t_f fixe ou libre.
- La fonction ϕ représente un coût ou critère terminal à payer à l'arrivée.
- C_0 et C_f sont des sous variétés différentiables de $\mathbb{R}^n$ (Intersection d'hyper surfaces $S_i = c_i^{-1}(0)$) et on note $\frac{\partial C}{\partial x}^T = (\frac{\partial c_1}{\partial x}, \cdots, \frac{\partial c_p}{\partial x})$

Théorème du minimum (cas temps invariant) :

Posons

$$H(x, \lambda, u) = \lambda^T f(x, u) + L(x, u)$$

la fonction Hamiltonienne associée au problème de commande.

Théorème

Si $\hat{u}(.)$ et $\hat{x}(.)$ sont respectivement une commande optimale et la trajectoire correspondantes pour le problème de commande, alors il existe une fonction absolument continue $\hat{\lambda}(.)$ telle que $(\hat{x}, \hat{\lambda}, \hat{u})$ vérifient :

1. *les équations canoniques de Hamilton* $\dot{x} = \frac{\partial H}{\partial \lambda}$, $\dot{\lambda} = -\frac{\partial H}{\partial x}$
2. *les conditions du minimum :*

$$H(\hat{x}, \hat{\lambda}, \hat{u}) = \inf_{u \in U} H(\hat{x}, \hat{\lambda}, u)$$

3. *les conditions de transversalité :* $\exists\ \alpha_0 \in \mathbb{R}^{p_0}$ *et* $\alpha_f \in \mathbb{R}^{p_f}$ *:*

$$\lambda(t_0) = \sum_{i=1}^{p_0} \frac{\partial c_{0,i}}{\partial x} \alpha_{0,i} \quad \lambda(t_f) = \sum_{i=1}^{p_f} \frac{\partial c_{i,f}}{\partial x} \alpha_{f,i} + \frac{\partial \phi(x(t_f))}{\partial x}$$

Autrement dit, λ est une combinaison linéaire des gradients des contraintes aux extrémités.

Démonstration : admise (à voir comme la version en temps continu de Kuhn et Tucker)

Remarque

Les conditions énoncées sont des conditions nécessaires pour qu'une commande et la trajectoire correspondante soient optimales. Elles n'en garantissent pas l'existence.

Remarque

Les conditions de transversalité associées aux conditions aux extrémités sur x fournissent $2n$ conditions réparties en t_0 et t_f.

Cas particuliers :

- si $x(t_0) = x_0$, $\lambda(t_0)$ est libre car $\frac{\partial C_0}{\partial x} = \frac{\partial (x - x_0)}{\partial x} = I_d$
- si $x(t_f) = x_f$, $\lambda(t_f)$ est libre
- si $x(t_f)$ est libre (pas de contrainte), alors

$$\lambda(t_f) = \begin{cases} \frac{\partial \phi(x(t_f))}{\partial x} & \text{si critère terminal} \\ 0 \text{ sinon} \end{cases}$$

Comment appliquer le théorème

Exercice : Problème en temps minimum

- Soit le système inertiel d'équation :

$$\ddot{z} = u$$

- La commande u est contrainte :

$$|u| \leq 1$$

- On veut transférer le système de l'état initial : $z(0) = 10, \dot{z}(0) = 0$ à l'état final $z(t_f) = 0,\ \dot{z}(t_f) = 0$ en temps minimum.

Comment appliquer le théorème

La démarche est la suivante :

1. Mettre les équations du système sous la forme d'une équation d'état normalisée ($\dot{x} = f(x, u)$)

   ici on posera $x_1 = z, x_2 = \dot{z}$. On obtient alors :

   $$\dot{x}_1 = x_2$$
   $$\dot{x}_2 = u$$

2. Mettre le critère sous forme intégrale. On cherche une commande en temps minimum, on veut donc minimiser t_f soit le critère :

   $$J = \int_0^{t_f} dt$$

3. Ecrire l'hamiltonien du problème :

   $$H(x, \lambda, u) = 1 + \lambda_1 x_2 + \lambda_2 u$$

4. Minimiser H.

   Ici la solution est simple : H étant une fonction affine de u ses extrema sont sur la frontière donc

   $$\hat{u} = -signe(\lambda_2).$$

Comment appliquer le théorème

5. On utilise maintenant les équations de Hamilton pour trouver des informations sur le comportement du vecteur adjoint :

$$\dot{\lambda}_1 = -\frac{\partial H}{\partial x_1} = 0;\ \lambda_1 \text{ est constant}$$
$$\dot{\lambda}_2 = -\frac{\partial H}{\partial x_2} = -\lambda_1;\ \lambda_2(t) = -\lambda_1 t + \lambda_2(0)$$

6. On en déduit que $\lambda_2(t)$ change au plus une fois de signe sur l'intervalle $(0, t_f)$.
   Les commandes satisfaisant les conditions nécessaires d'optimalité sont donc de la forme :

$$u(t) = \pm 1, \quad t \in (0, t_c)$$
$$u(t) = \mp 1, \quad t \in (t_c, t_f)$$

   avec $t_0 \leq t_c \leq t_f$.

Comment appliquer le théorème

7. Il faut maintenant chercher dans cet ensemble de commandes celles qui satisfont les conditions aux limites. Comme l'état initial et l'état final sont imposés, il n'y a aucune information sur le vecteur adjoint.
8. Ici, à partir des C.I. du problème et en considérant l'évolution de la vitesse, on déduit immédiatement que la vitesse doit être < 0, donc la commande optimale est :

$$\hat{u}(t) = -1 \ pour \ t \in (0, t_c) \ ; \hat{u}(t) = 1 \ pour \ t \in (t_c, t_f)$$

Comment appliquer le théorème

9. On peut alors résoudre les équations et en déduire t_f.
   Avant $t < t_c$

$$x_2(t) = -t \quad x_1(t) = -\frac{t^2}{2} + 10 \Rightarrow x_1 = -\frac{x_2^2}{2} + 10$$

   A $t = t_c$

$$x_2(t_c) = -t_c \quad x_1(t_c) = -\frac{t_c^2}{2} + 10$$

   Après $t > t_c$

$$x_2(t) = t - 2t_c \quad x_1(t) = \frac{t^2}{2} - 2t_c t + t_c^2 + 10$$

   A $t = t_f$

$$x_2(t_f) = t_f - 2t_c \quad x_1(t_f) = \frac{t_f^2}{2} - 2t_c t_f + t_c^2 + 10$$

   d'où le résultat :

$$x_2(t_f) = 0 \Rightarrow t_c = \frac{t_f}{2} \qquad x_1(t_f) = \frac{t_f^2}{2} - t_f^2 + \frac{t_f^2}{4} + 10$$

$$x_1(t_f) = 0 \Rightarrow t_f = \sqrt{40} \; t_c = \sqrt{10}$$

   Soit dans le plan de phase :

$$pour \; t \in (0, t_c) \; x_1 = -\frac{x_2^2}{2} + 10 \quad pour \; t \in (t_c, t_f) \; x_1 = \frac{x_2^2}{2}$$

Exemple d'Algorithme

Problème avec coût terminal :

$$\dot{x} = f(x,u), \quad x(0) = x_0 \quad J = \phi(x(T)) + \int_0^T L(x,u)dt$$

Algorithme :

On construit une suite $u^i|_{[0\ T]}$, $i = 0, 1, ...$ qui converge vers les CN

- $u^0|_{[0\ T]}$ donné et quelconque.

  Pour $i = 0, 1, ...$

  1. Intégration sur $[0\ T]$, du système :
  $$\dot{x}^i = f(x^i, u^i)$$
  2. Condition de transversalité en T :
  $$\lambda(T)^i = \frac{\partial \phi}{\partial x}(x(T)^i)$$
  3. Intégration sur $[T\ 0]$ du système adjoint :
  $$\dot{\lambda}^i = -\frac{\partial H(x^i, u^i, \lambda^i)}{\partial x}$$
  4. Mise à jour de u :
  $$u^{i+1}|_{[0\ T]} = u^i|_{[0\ T]} - \rho_i \frac{\partial H(x^i, u^i, \lambda^i)}{\partial u}|_{[0\ T]} \text{ (gradient ou autre chose)}$$
  5. $i \to i+1$ tant que test d'arrêt non satisfait : $\| \frac{\partial H(x^i, u^i, \lambda^i)}{\partial u} \| < \epsilon$

  Fin Pour

La commande LQ (système linéaire, critère quadratique) Formulation continue

- Soit le système linéaire :

$$\dot{x} = Ax + Bu$$
$$x(0) = x_0,\ x(t_f) \text{ libre },\ x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m$$

- On cherche la commande qui minimise le critère quadratique sur l'horizon t_f fixé :

$$J(u) = \frac{1}{2}\int_0^{t_f} [x^T Q x + u^T R u]dt + \frac{1}{2}x(t_f)^T P_f x(t_f)$$

où $Q = Q^T \geq 0$, $R = R^T > 0$, $P_f = P_f^T \geq 0$.

- On écrira $Q = C^T C$, et on supposera (A, B) commandable et (A, C) observable.

Résolution par le théorème de Pontriaguine

Equations générales :

On applique le théorème de Pontriaguine en présence d'un critère terminal.

- L'hamiltonien du problème est :

$$H = \lambda^T(Ax + Bu) + \frac{1}{2}(x^T Qx + u^T Ru)$$

- La commande optimale est déterminée par la CN : $\frac{\partial H}{\partial u} = 0$, donc $Ru + B^T\lambda = 0$ et

$$\hat{u} = -R^{-1}B^T\lambda$$

- l'équation adjointe est :

$$\dot{\lambda} = -\frac{\partial H}{\partial x} = -A^T\lambda - Qx$$

- Soit le système différentiel

$$\dot{x} = Ax - BR^{-1}B^T\lambda$$
$$\dot{\lambda} = -Qx - A^T\lambda$$

soit encore :

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = M \begin{bmatrix} x \\ \lambda \end{bmatrix}$$

La matrice M est appelée :" matrice hamiltonienne du système".

En résolvant formellement le système hamiltonien :

$$\begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix} = \exp(Mt) \begin{bmatrix} x_0 \\ \lambda_0 \end{bmatrix} = \begin{bmatrix} \phi_{11}(t) & \phi_{12}(t) \\ \phi_{21}(t) & \phi_{22}(t) \end{bmatrix} \begin{bmatrix} x_0 \\ \lambda_0 \end{bmatrix}$$

avec les conditions aux extrémités :

$$x(t_0) = x_0$$

$$\lambda(t_f) = \frac{\partial(\frac{1}{2}x^T P_f x)}{\partial x} = P_f x(t_f)$$

car $x(t_f)$ est libre,
on voit (après quelques calculs !) que la solution générale peut se mettre sous la forme :

$$\lambda(t) = P(t)x(t) \quad P(t_f) = P_f$$

Le calcul donne :

$$P(t) = (\phi_{22}(t_f - t) - P_f\phi_{12}(t_f - t))^{-1}(P_f\phi_{11}(t_f - t) - \phi_{21}(t_f - t))$$

Solution : équation de Riccati :

- Cherchons l'équation différentielle vérifiée par $P(t)$ par dérivation de $\lambda(t) = P(t)x(t)$

$$\begin{aligned}
\dot{\lambda} &= \dot{P}(t)x(t) + P(t)\dot{x}(t) \\
-Qx - A^T\lambda &= \dot{P}x + P(Ax + Bu) \\
-Qx - A^TPx &= \dot{P}x + PAx - PBR^{-1}B^TPx, \quad \forall x \in \mathbb{R}^n
\end{aligned}$$

- On en déduit que P vérifie l'équation différentielle, dite équation différentielle de Riccati

$$\begin{cases} -\dot{P} & = PA + A^TP - PBR^{-1}B^TP + Q \\ P(T) & = P_f \end{cases}$$

- en substituant $u(t) = -R^{-1}B^T\lambda(t) = -R^{-1}B^TPx(t)$ et en utilisant l'équation de Riccati on trouve :

$$\begin{aligned}
\hat{J}(x_0) &= \frac{1}{2}\int_0^{t_f} [x^TQx + x^TPBR^{-1}B^TPx]dt + \frac{1}{2}x(t_f)^TP_fx(t_f) \\
&= -\frac{1}{2}\int_0^{t_f} \frac{dx^TPx}{dt}dt + \frac{1}{2}x(t_f)^TP_fx(t_f) = \frac{1}{2}x_0^TP(0)x_0
\end{aligned}$$

Bilan

Résumons les résultats obtenus.

- Equation différentielle de Riccati

$$\begin{cases} -\dot{P} & = PA + A^T P - PBR^{-1}B^T P + Q \\ P(T) & = P_f \end{cases}$$

- Loi de feedback dynamique et critère optimaux :

$$\hat{u}(t,x) = -K(t)x(t) \text{ avec } K(t) = R^{-1}B^T P(t)$$

$$\hat{J}(x_0) = \frac{1}{2} x_0^T P(t) x_0$$

La valeur du critère est positive quel que soit x, t donc $P > 0$

Comportement asymptotique du régulateur LQ :

On cherche à résoudre :

$$\dot{x} = Ax + Bu \quad x(0) = x_0 \qquad \min_u \frac{1}{2} \int_0^{+\infty} x^T Q x + u^T R u \; dt$$

Théorème

Lorsque (A,B) est commandable et (A,C) observable,

1. *La commande optimale est un retour d'état :*

$$u = -Kx \text{ avec } K = R^{-1} B^T P$$

où P est l'unique solution symétrique et définie positive ($P = P^T > 0$) de l'équation algébrique de Riccati,

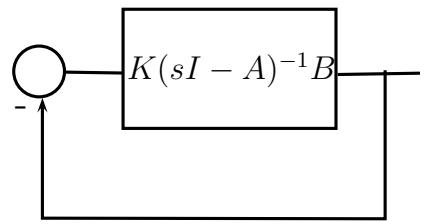
$$PA + A^T P - PBR^{-1} B^T P + Q = 0$$

2. *Le système ainsi régulé est asymptotiquement stable.*
3. *Le coût optimal est déterminé par : $\hat{J}(x_0) = V(x_0)$ où V est la fonction de Lyapunov :*

$$V(x) = \frac{1}{2} x^T P x$$

Interprétation cas mono entrée : Marges de phase, marge de gain

- On considère les systèmes mono-entrée ; $u \in \mathbb{R}$
- Le critère s'écrit : $J = \int_0^\infty (x^T Q x + u^2) dt$ (R=1)
- Le régulateur LQ conduit à un retour d'état -Kx selon le schéma.



- Cette commande est donc équivalente à une fonction de transfert

$$G(s) = K(sI - A)^{-1}B$$

bouclée par un retour unitaire.

Robustesse du régulateur LQ

- En utilisant l'expression du gain de retour, l'équation de Riccati peut se mettre sous la forme

$$-PA - A^T P + K^T RK = Q$$

- En ajoutant $0 = sP - sP$, on obtient :

$$P(sI - A) + (-sI - A^T)P + K^T RK = Q$$

- Ce qui en multipliant à droite par $(sI - A)^{-1}B$ et à gauche par $B^T(-sI - A^T)^{-1}$ donne :

$$B^T(-sI-A^T)^{-1}PB+B^TP(sI-A)^{-1}B+B^T(-sI-A^T)^{-1}K^TRK(sI-A)^{-1}B$$
$$= B^T(-sI - A^T)^{-1}Q(sI - A)^{-1}B$$

Robustesse du régulateur LQ

- En utilisant : $RK = B^T P$, en ajoutant R de chaque coté et en factorisant, on a l'égalité caractéristique du régulateur LQ. :

$$[I+B^T(-sI - A^T)^{-1}K^T]R[I+K(sI - A)^{-1}B] = R+B^T(-sI-A^T)^{-1}Q(sI-A)^{-1}B$$

- De cette équation, en prenant $s = j\omega$ on tire une inéquation fondamentale dans l'étude de la robustesse puisque $Q \geq 0$:

$$[I + G(-j\omega)^T]R[I + G(j\omega)] \geq R > 0$$

avec $G(s) = K(sI - A)^{-1}B$

Marge de phase, marge de gain

- L'égalité fondamentale appliquée à ce cas mono-entrée, $R = 1$ donne :

$$[1 + G(s)][1 + G(-s)] = 1 + B^T(-sI - A^T)^{-1}Q(sI - A)^{-1}B$$

soit dans le domaine fréquentiel :

$$|1 + G(j\omega)|^2 = 1 + B^T(-j\omega I - A^T)^{-1}Q(j\omega I - A)^{-1}B \geq 1, \quad \forall\omega$$

- Traduit dans le lieu de Nyquist, cette inégalité montre que la courbe de Nyquist est extérieure au cercle de centre -1 et de rayon 1.
- Les marges de phase et de gain s'en déduisent :
  $\Delta G = [1/2,\ +\infty],\ \Delta\phi \geq 60°$