Algorithmes I

(P. Carpentier)

24 mars 2017

Plan du cours

Minimisation sans contrainte et algorithme

Problème d'optimisation sans contrainte.

$$\min_{u \in \mathbb{U}} J(u).$$

Si J est différentiable, on a la condition d'optimalité (**KKT**) :

$$\nabla J(u^\sharp) = 0.$$

Notion d'algorithme.

Un algorithme est une application $\mathcal{A}$ de l'espace $\mathbb{U}$ dans lui-même.

Le déroulement de l'algorithme à partir d'un point initial $u^{(0)} \in \mathbb{U}$ consiste donc à utiliser de manière itérative cette application :

$$u^{(k+1)} = \mathcal{A}(u^{(k)}) \ , \quad k = 0, 1, \ldots$$

L'algorithme converge si la suite $\{u^{(k)}\}_{k \in \mathbb{N}}$ converge dans $\mathbb{U}$.

Vitesse de convergence

Un algorithme d'optimisation engendre une suite $\{u^{(k)}\}_{k\in\mathbb{N}}$ qui, *dans les bons cas*, converge vers une solution $u^\sharp$ du problème.

Convergence en quotient

- Convergence q-linéaire :

$$\exists r \in [0,1[,\ \exists K_r,\ \forall k \geq K_r,\ \left\|u^{(k+1)} - u^\sharp\right\| \leq r\left\|u^{(k)} - u^\sharp\right\|.$$

- Convergence q-superlinéaire :

$$\forall r > 0,\ \exists K_r,\ \forall k \geq K_r,\ \left\|u^{(k+1)} - u^\sharp\right\| \leq r\left\|u^{(k)} - u^\sharp\right\|.$$

- Convergence q-quadratique :

$$\exists C > 0,\ \forall k \geq 1,\ \left\|u^{(k+1)} - u^\sharp\right\| \leq C\left\|u^{(k)} - u^\sharp\right\|^2.$$

"Le nombre de chiffres significatifs corrects **double** à chaque itération."

Schéma de principe d'un programme d'optimisation

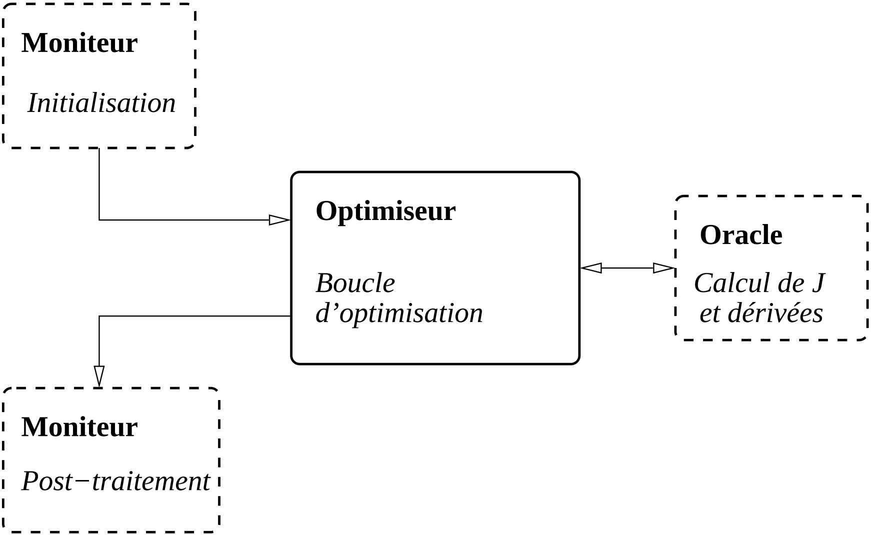


FIGURE: Oracle en communication directe

Quelques classes de méthodes d'optimisation (1)

Méthodes pour l'optimisation globale.

- Optimisation polynomiale, programmation semi-définie.
- Algorithme génétique, recuit simulé, essaim de particules...

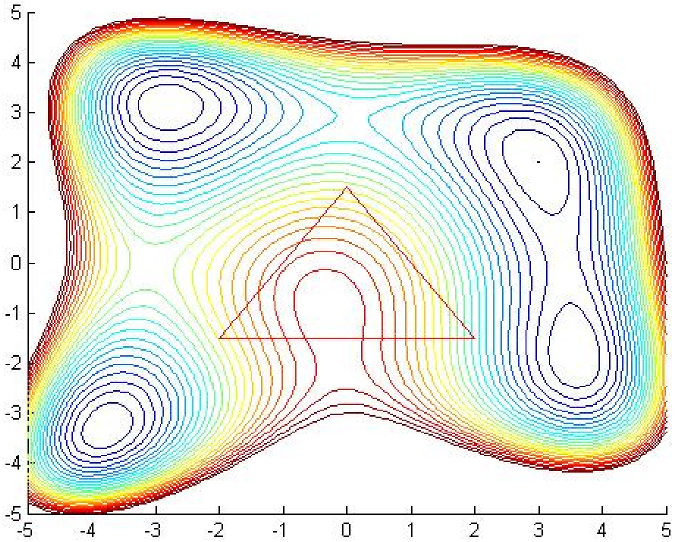
Méthodes n'utilisant pas les dérivées.

- Méthodes de dichotomie :
  - $J : \mathbb{R} \longrightarrow \mathbb{R}$,
  - on suppose J unimodale ($J \downarrow$ si $u \leq u^\sharp$ et $J \uparrow$ si $u \geq u^\sharp$),
  - on divise l'intervalle de recherche par 2 à chaque itération, au prix de 2 évaluations de J par itération (voir aussi la recherche par la section d'or).
- Algorithme de Nelder-Mead :
  - $J : \mathbb{R}^n \longrightarrow \mathbb{R}$,
  - on se donne $n+1$ points $(u_1, \ldots, u_{n+1})$ que l'on ordonne :

$$J(u_1) \leq \ldots \leq J(u_{n+1}),$$

  - on remplace à chaque itération le pire point u_{n+1} en déformant le simplexe des points restants par étirement ou contraction.

Algorithme de Nelder-Mead et fonction de Himmelblau



Méthodes à régions de confiance.

- On se donne un modèle de la variation de J autour de $u^{(k)}$:

$$\phi^{(k)}(v) = {g^{(k)}}^\top v + \frac{1}{2} v^\top H^{(k)} v,$$

- et un voisinage dans lequel ce modèle est réputé « bon » :

$$B(0, \Delta^{(k)}) = \left\{ v \in \mathbb{U},\ \|v\| \leq \Delta^{(k)} \right\}.$$

- On résout le sous-problème quadratique :

$$\min_{v \in B(0,\Delta^{(k)})} \phi^{(k)}(v) \quad \rightsquigarrow \quad v^{(k)}.$$

- On calcule la concordance du modèle :

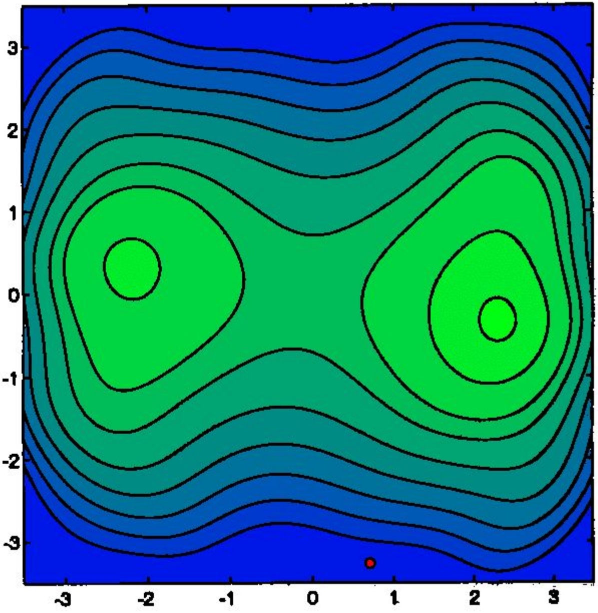
$$\rho^{(k)} = \frac{J(u^{(k)} + v^{(k)}) - J(u^{(k)})}{\phi^{(k)}(v^{(k)})}.$$

  - Si la concordance est « mauvaise », on diminue $\Delta^{(k)}$.
  - Sinon, on met à jour le point courant :

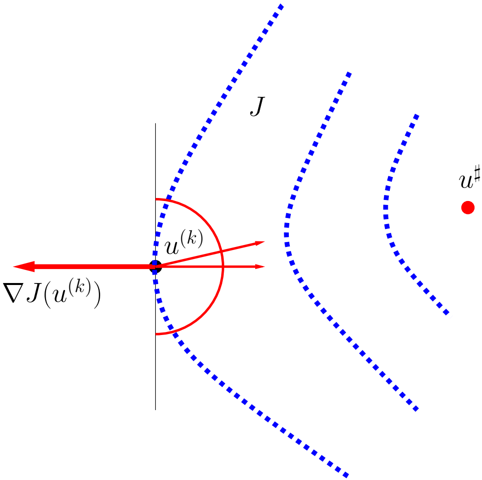
    $$u^{(k+1)} = u^{(k)} + v^{(k)},$$

    on met à jour le rayon de confiance $\Delta^{(k)}$ et on itère.

Régions de confiance et fonction de Himmelblau



Méthodes à directions de descente.



Ce sont les plus classiques, que l'on va maintenant développer.

Principe des méthodes à directions de descente

Problème d'optimisation différentiable sans contrainte :

$$\min_{u \in \mathbb{U}} J(u).$$

Direction de descente $d^{(k)}$ **en** $u^{(k)}$: $\left\langle \nabla J(u^{(k)}) \, , d^{(k)} \right\rangle < 0$.

$$\rightsquigarrow \exists \alpha^{(k)} > 0, \; J(u^{(k)} + \alpha^{(k)} d^{(k)}) < J(u^{(k)}).$$

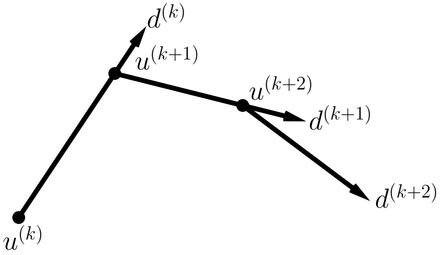


FIGURE: Algorithme à direction de descente : $u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)}$

Description d'une itération de la méthode

Itération k d'un algorithme de type direction de descente

On dispose d'un point $u^{(k)} \in \mathbb{U}$ au début de l'itération.

1. Test d'arrêt :
   si $\left\|\nabla J(u^{(k)})\right\| < \epsilon$, convergence de l'algorithme.
2. Direction de descente :
   calculer une direction de descente $d^{(k)}$ au point $u^{(k)}$.
3. Recherche linéaire :
   déterminer un pas $\alpha^{(k)} > 0$ tel que J "diminue suffisamment".
4. Rebouclage :
   calculer $u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)}$ et passer à l'itération $k+1$.

Éviter si possible un test d'arrêt sur les variations de u ou de J...

Problématique de la direction de descente

Méthodes à directions de descente.

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)} \ .$$

Déterminer la direction $d^{(k)}$.

- **Algorithme du gradient** :

$$d^{(k)} = -\nabla J(u^{(k)}).$$

Le plus simple, le plus lent (au voisinage de $u^\sharp$). Pourtant...

- **Algorithmes de gradient conjugué** :

$$d^{(k)} = \begin{cases} -\nabla J(u^{(0)}) & \text{si } k = 0 \\ -\nabla J(u^{(k)}) + \beta^{(k)} d^{(k-1)} & \text{si } k \geq 1 \end{cases}.$$

Différents algorithmes suivant l'expression de $\beta^{(k)} \in \mathbb{R}$.

- **Algorithme de Newton** :

$$d^{(k)} \text{ solution de : } \nabla^2 J(u^{(k)}) d + \nabla J(u^{(k)}) = 0.$$

Il faut que la matrice hessienne $\nabla^2 J(u^{(k)})$ soit inversible.

- **Algorithmes de quasi-Newton** :

$$d^{(k)} = -W^{(k)} \nabla J(u^{(k)}).$$

$W^{(k)}$ *: approximation de l'inverse du hessien $\nabla^2 J(u^{(k)})$.*

Méthodes à directions de descente.

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)} \ .$$

Déterminer le pas $\alpha^{(k)}$ dont on se déplace dans la direction $d^{(k)}$.

Recherche linéaire : considérations générales

À l'itération k, on détermine un pas $\alpha^{(k)}$ indiquant de combien on veut se déplacer dans la direction $d^{(k)}$ à partir d'un point $u^{(k)}$. On s'intéresse donc à la fonction $\varphi_k : \mathbb{R} \longrightarrow \mathbb{R}$ définie par

$$\varphi_k(\alpha) = J(u^{(k)} + \alpha d^{(k)}) \quad \rightsquigarrow \quad \nabla\varphi_k(0) = \left\langle \nabla J(u^{(k)})\,, d^{(k)} \right\rangle.$$

Recherche linéaire exacte :

$$\min_{\alpha \geq 0} \varphi_k(\alpha).$$

Le pas optimal $\alpha_k^\sharp$ est appelé **pas de Cauchy**.
Il peut être coûteux à calculer, et n'est pas toujours intéressant...

Recherche linéaire approchée : on poursuit 2 objectifs :

1. faire décroître J « significativement »,
2. empêcher le pas $\alpha^{(k)}$ d'être « trop petit ».

(Car avec $\alpha^{(k)} = \dfrac{\epsilon}{2^k \|d^{(k)}\|}$, on converge dans la boule $B(u^{(0)}, \epsilon)$!)

Recherche linéaire approchée.

Premier objectif : J doit décroître autant qu'une fraction $\omega_1 \in]0,1[$ de ce que ferait le modèle linéaire de J en $u^{(k)}$:

$$J(u^{(k)} + \alpha^{(k)} d^{(k)}) \leq J(u^{(k)}) + \omega_1 \alpha^{(k)} \left\langle \nabla J(u^{(k)}) \, , d^{(k)} \right\rangle ,$$

inégalité qui s'écrit sous la forme équivalente :

$$\varphi_k(\alpha^{(k)}) \leq \varphi_k(0) + \omega_1 \alpha^{(k)} \nabla \varphi_k(0).$$

Cette condition s'appelle la règle d'Armijo.

Elle est toujours vérifiée si le pas $\alpha^{(k)}$ est pris suffisamment petit. (On a $\nabla \varphi_k(0) < 0$ puisque $d^{(k)}$ est une direction de descente).

En pratique, la constante ω_1 est choisie « petite » : $\omega_1 = 10^{-3}$.

Recherche linéaire : règle de Wolfe (2)

Première condition de Wolfe : $\varphi_k(\alpha^{(k)}) \leq \varphi_k(0) + \omega_1 \alpha^{(k)} \nabla \varphi_k(0)$.

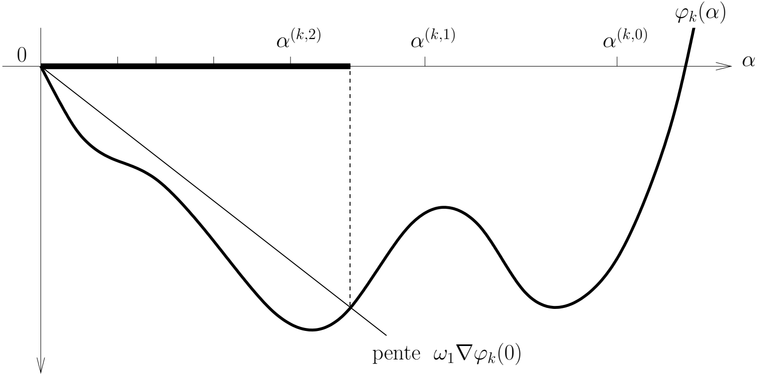


FIGURE: Plage des pas admissibles pour la règle d'Armijo

La satisfaction de cette condition se fait par "backtracking" : on part d'un pas $\alpha^{(k,0)}$ suffisamment grand, on se donne $\tau \in]0,1[$ et on calcule le plus petit indice ℓ tel que le pas $\alpha^{(k,\ell)} = \tau \alpha^{(k,\ell-1)}$ vérifie la condition.

Recherche linéaire approchée.

Second objectif : pour empêcher le pas $\alpha^{(k)}$ d'être trop petit, on se donne une deuxième constante ω_2 telle que $0 < \omega_1 < \omega_2 < 1$ et on impose la condition :

$$\left\langle \nabla J(u^{(k)} + \alpha^{(k)} d^{(k)}) \, , d^{(k)} \right\rangle \geq \omega_2 \left\langle \nabla J(u^{(k)}) \, , d^{(k)} \right\rangle ,$$

inégalité qui s'écrit sous la forme équivalente :

$$\nabla \varphi_k(\alpha^{(k)}) \geq \omega_2 \nabla \varphi_k(0).$$

Cette condition n'est pas vérifiée pour des pas $\alpha^{(k)}$ trop petits (on rappelle que $\nabla \varphi_k(0) < 0$).

En pratique, la constante ω_2 est choisie proche de 1 : $\omega_2 = 0.99$.

Règle de Wolfe :

$$J(u^{(k)} + \alpha^{(k)} d^{(k)}) - J(u^{(k)}) \leq \omega_1 \alpha^{(k)} \left\langle \nabla J(u^{(k)}), d^{(k)} \right\rangle,$$

$$\left\langle \nabla J(u^{(k)} + \alpha^{(k)} d^{(k)}), d^{(k)} \right\rangle \geq \omega_2 \left\langle \nabla J(u^{(k)}), d^{(k)} \right\rangle.$$

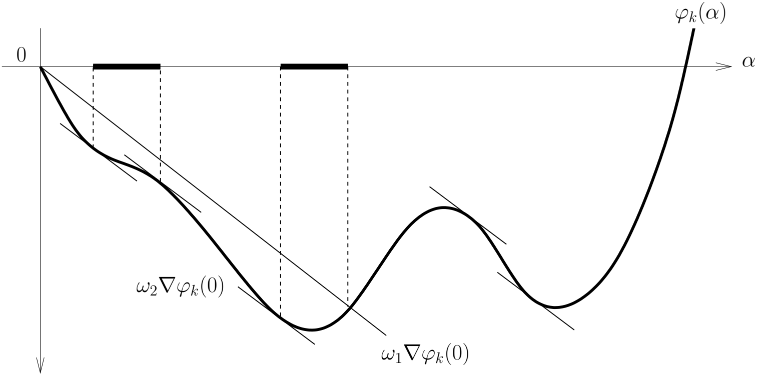


FIGURE: Plages des pas admissibles pour la règle de Wolfe

Recherche linéaire : algorithme de Fletcher-Lemaréchal

Initialisation

On pose $\underline{\alpha} = 0$ et $\overline{\alpha} = +\infty$ et on se donne $\alpha^{(k,0)} \in]\underline{\alpha}, \overline{\alpha}[$.

Iteration ℓ

- Si $\alpha^{(k,\ell)}$ ne vérifie pas la 1-ère condition de Wolfe :
  - on diminue la borne *supérieure* : $\overline{\alpha} = \alpha^{(k,\ell)}$,
  - on choisit un nouveau pas : $\alpha^{(k,\ell+1)} = \frac{1}{2}(\underline{\alpha} + \overline{\alpha})$
  - on effectue une nouvelle itération.
- Sinon :
  - si $\alpha^{(k,\ell)}$ ne vérifie pas la 2-ème condition de Wolfe :
    - on augmente la borne *inférieure* : $\underline{\alpha} = \alpha^{(k,\ell)}$,
    - on choisit un nouveau pas :

$$\begin{aligned} \alpha^{(k,\ell+1)} &= 2\underline{\alpha} && \text{si } \overline{\alpha} = +\infty, \\ \alpha^{(k,\ell+1)} &= \tfrac{1}{2}(\underline{\alpha} + \overline{\alpha}) && \text{sinon,} \end{aligned}$$

    - on effectue une nouvelle itération,
  - sinon, le pas $\alpha^{(k,\ell)}$ vérifie la règle de Wolfe.

À préciser : choix du pas initial $\alpha^{(k,0)}$ *et test d'arrêt pour cet algorithme.*

Recherche linéaire : convergence et mise en œuvre

Convergence de l'algorithme de Fletcher-Lemaréchal

Soit $d^{(k)}$ une direction de descente. On fait l'hypothèse que φ_k est dérivable et bornée inférieurement, et que $0 < \omega_1 < \omega_2 < 1$. Alors, l'algorithme de Fletcher-Lemaréchal trouve un pas $\alpha^{(k)}$ vérifiant la règle de Wolfe en un nombre fini d'itérations.

Mise en œuvre de l'algorithme de Fletcher-Lemaréchal

- Choix du pas initial.
  - Algorithme de type Newton : pas unité :
  $$\alpha^{(k,0)} = 1.$$
  - Algorithme de type gradient : pas de Fletcher :
  $$\alpha^{(k,0)} = -2\frac{\Delta_k}{\left\langle \nabla J(u^{(k)}) \, , d^{(k)} \right\rangle},$$
  où Δ_k est la *décroissance attendue* du critère.
- Critère de convergence.
  Plutôt que de considérer l'écart $\left|\alpha^{(k,\ell+1)} - \alpha^{(k,\ell)}\right|$, on se basera sur $\left\|u^{(k,\ell+1)} - u^{(k,\ell)}\right\|$ pour arrêter l'algorithme.