HMMA307

Optimisation convexe avec/sans contrainte(s)

Cours: Joseph Salmon *Scribes: MASSOL Océane et Cheikh Tidiane Mbaye*

1 Optimisation convexe sans contrainte

Theorem.

$f : \mathbb{R}^d \longrightarrow \mathbb{R}^1$
$x \longrightarrow f(x)$
Si f est C^1 et convexe, alors $x^* \in \operatorname{argmin}_{x \in \mathbb{R}^d} f \Leftrightarrow \nabla f(x^*) = 0$.

Lemme 1.1.

$f : \mathbb{R}^d \longrightarrow \mathbb{R}$
Si f est C^1, alors : f est convexe $\Leftrightarrow \forall\ x_1,\ x_2 \in \mathbb{R}^d \times \mathbb{R}^d,\ f(x_2) \geqslant f(x_1) + \langle x_2 - x_1; \nabla f(x_1) \rangle$.

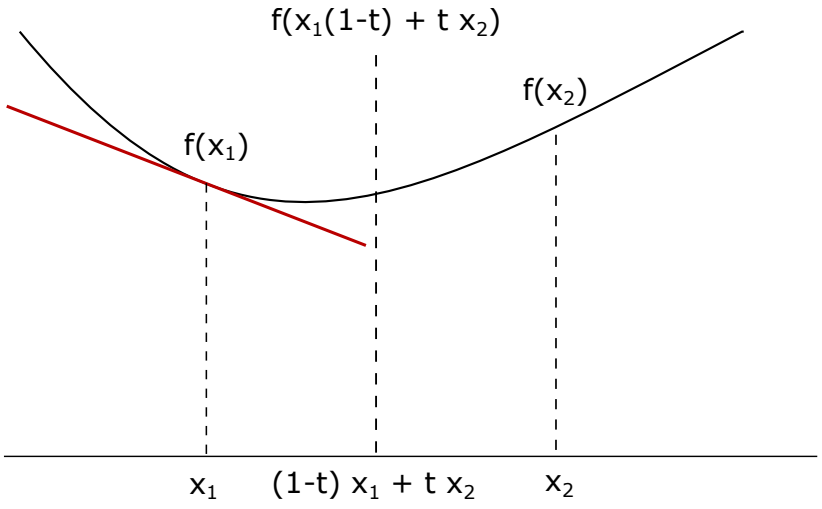

Figure 1: Fonction convexe et une de ses tangentes

Preuve. (Lemme)

Partie 1 :

Pour tout $t \in]0,1[$, nous avons :

$$\begin{aligned} f((1-t)x_1 + tx_2) &= f(x_1 + t(x_2 - x_1)) \\ &\leqslant (1-t)f(x_1) + tf(x_2) \text{ (convexité)} \\ \Leftrightarrow f(x_1 + t(x_2 - x_1)) - f(x_1) &\leqslant t(f(x_2) - f(x_1)) \\ \Leftrightarrow \frac{f(x_1 + t(x_2 - x_1)) - f(x_1)}{t} &\leqslant (f(x_2) - f(x_1)) \text{ (car l'on a divisé par t.)} \end{aligned} \tag{1}$$

Lorsque t $\longrightarrow$ 0, on a : $\langle \nabla f(x_1); x_2 - x_1 \rangle \leqslant f(x_2) - f(x_1)$.

Partie 2 :

$\forall\ x_1,\ x_2$ et $t \in [0,1]$, on a :

$$f(x_1) \geqslant f(t(x_1 - x_2) + x_2) - \langle t(x_1 - x_2) + (x_2 - x_1); \nabla f(t(x_1 - x_2) + x_2) \rangle. \tag{2}$$

$$f(x_2) \geqslant f(t(x_1 - x_2) + x_2) + \langle x_2 - x_2 - t(x_1 - x_2); \nabla f(t(x_1 - x_2) + x_2) \rangle. \tag{3}$$

Nous multiplions l'équation (2) par t et l'équation (3) par (1-t), puis nous les sommons.
Nous obtenons le résultat suivant :

$$\begin{aligned} tf(x_1) + (1-t)f(x_2) &\geqslant tf(t(x_1 - x_2) + x_2) + (1-t)f(t(x_1 - x_2) + x_2)(-t(1-t) + (1-t)t) \\ &\quad \langle x_2 - x_2 - t(x_1 - x_2); \nabla f(t(x_1 - x_2) + x_2) \rangle \\ &\geqslant f(t(x_1 - x_2) + x_2) \\ &= f(tx_1 + (1-t)x_2). \end{aligned} \tag{4}$$

Preuve. (Théorème)

Partie 1 :

On a $x^* \in \operatorname{argmin}_{x\in\mathbb{R}^d} f(x)$.

On utilise la formule de Taylor à l'ordre 1 : $f(x) = f(x^*) + \langle \nabla f(x^*); x - x^* \rangle + o(\| x - x^* \|)$.

Ainsi, $\forall\ x$, nous avons :

$$\begin{aligned} & f(x) \geqslant f(x^*) \\ \Leftrightarrow\ & f(x^*) + \langle \nabla f(x^*); x - x^* \rangle + o(\| x - x^* \|) \geqslant f(x^*) \\ \Leftrightarrow\ & \langle \nabla f(x^*); x - x^* \rangle + o(\| x - x^* \|) \geqslant 0 \end{aligned} \tag{5}$$

$\forall\ h$, on a :

$$\begin{aligned} & \langle \nabla f(x^*); h \rangle + o(\| h \|) \geqslant 0 \\ & \Leftrightarrow \langle \nabla f(x^*); \frac{h}{\| h \|} \rangle + o(1) \geqslant 0 \text{ , avec } h = -\nabla f(x^*) \\ & \Leftrightarrow -\langle \nabla f(x^*); \frac{\nabla f(x^*)}{\| \nabla f(x^*) \|} \rangle + o(1) \geqslant 0 \\ & \Leftrightarrow -\| \nabla f(x^*) \| + o(1) \geqslant 0 \\ & \Leftrightarrow o(1) \geqslant \ \| \nabla f(x^*) \| . \end{aligned} \tag{6}$$

Cela implique que $\| \nabla f(x^*) \| = 0$.

Partie 2 :

$\forall\ x$, nous avons $f(x) \geqslant f(x^*) + \langle x - x^*; \nabla f(x^*) \rangle$ d'après le lemme 1.1.
Or, $\langle x - x^*; \nabla f(x^*) \rangle = 0$.
Ainsi, on a donc $\forall\ x : f(x) \geqslant f(x^*) \Rightarrow x^* \in \operatorname{argmin}_{x\in\mathbb{R}} f$.

2 Optimisation convexe avec contrainte(s)

On a $P^* = \min_{x\in\mathbb{R}^d} f_0(x)$, où P^* représente une valeur primale.

Nous avons les contraintes suivantes :

- $f_i(x) \leqslant 0,\ \forall i = 1, ..., n_1$.
- $h_j(x) = 0,\ \forall j = 1, ..., n_2$.

$f_0, f_i, f_j \ \forall i, \ \forall j : C^1$.
Dans la suite, on ne fera la théorie que pour f_0 convexe, f_i convexes et les h_j affines.

On note $\mathcal{F} = \{x \in \mathbb{R}^d : \forall i \ f_i(x) \leqslant 0, \forall j \ h_j(x) = 0\}$ l'ensemble faisable.

Definition. *Lagrangien*

On pose $x \in \mathbb{R}^d, \lambda \in \mathbb{R}^{n_1}, \mathcal{V} \in \mathbb{R}^{n_2}$.
$\mathcal{L}(x; \lambda; \mathcal{V}) = f_0(x) + \sum_{i=1}^{n_1} \lambda_i f_i(x) + \sum_{j=1}^{n_2} \mathcal{V}_j h_j(x)$, *avec* λ *et* $\mathcal{V}$ *les multiplicateurs de Lagrange.*

Definition. *Fonction duale*

Nous avons : $g(\lambda; \mathcal{V}) = \min_x \mathcal{L}(x; \lambda; \mathcal{V})$.

Remarque :

On a x fixé.
$(\lambda, \mathcal{V}) \longrightarrow \mathcal{L}(x; \lambda; \mathcal{V})$ est affine.
Le min (ou inf) de fonctions affines est concave : g est concave.

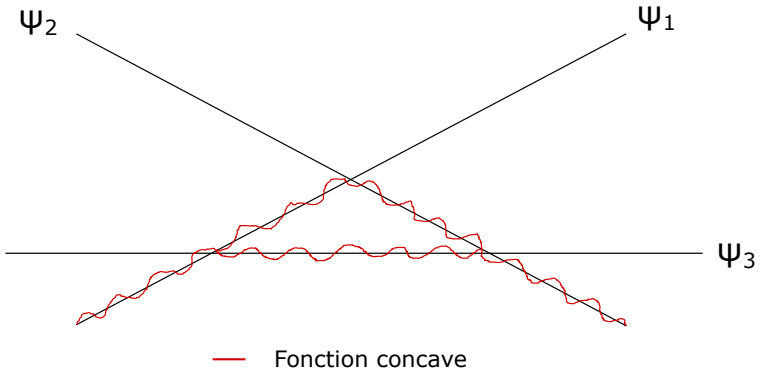

Figure 2: Fonctions affines

Remarque :

Si $\lambda \geqslant 0 \ (\forall i, \ \lambda_i \geqslant 0), \forall x \in \mathcal{F} : \mathcal{L}(x; \lambda; \mathcal{V}) \leqslant f_0(x)$.
$\Rightarrow g(\lambda; \mathcal{V}) \leqslant f_0(x)$
$\Rightarrow g(\lambda; \mathcal{V}) \leqslant p^*$

Probleme dual :

$d^* = \max_{\lambda, \mathcal{V}} \ g(\lambda, \mathcal{V})$, sous contrainte $\lambda \geqslant 0$.

$\forall \lambda \geqslant 0, \forall\ \mathcal{V}, \forall x \in \mathcal{F}$, on a $g(\lambda, \mathcal{V}) \leqslant d^* \leqslant p^* \leqslant f_0(x)$.

Dualite forte : c'est ce qu'il se passe lorsque $d^* = p^*$.

Saut dualite : $p^* - d^* \geqslant 0$ (dualité faible).

Theorem. *Shalev*

Si f_i est convexe et les h_j sont affines, alors $\exists\ \tilde{x} \in \mathbb{R}^d$ tel que $\forall i,\ f_i(\tilde{x}) < 0$ et $\forall j,\ h_j(\tilde{x}) = 0$, alors $p^* = d^*$ (dualité forte).

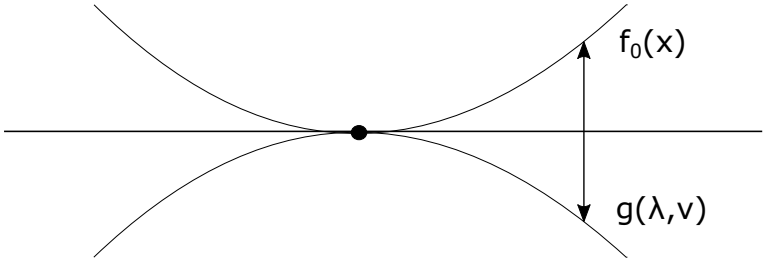

Figure 3: Illustration du théorème de Shalev

Conséquences de $\mathbf{d^* = p^*}$:

On note x^* la solution du primal et $(\lambda^*, \mathcal{V}^*)$ la solution duale.

$$\begin{aligned} f(x^*) &= p^* = d^* = g(\lambda^*, \mathcal{V}^*) \\ &= \inf_{x \in \mathcal{F}} f_0(x) + \sum_i \lambda_i^* f_i(x) + \sum_j \mathcal{V}_j^* h_j(x) \quad \text{(par définition de g).} \end{aligned} \tag{7}$$

$$\begin{aligned} f_0(x^*) &\leqslant f_0(x^*) + \sum_i \lambda_i^* f_i(x^*) + \sum_j \mathcal{V}_j^* h_j(x^*) \\ \Leftrightarrow 0 &\leqslant \sum_i \lambda_i^* f_i(x^*) \leqslant 0 \quad (\text{car } \sum_j \mathcal{V}_j^* h_j(x^*) = 0) \end{aligned} \tag{8}$$

Nous avons donc $\forall i,\ \lambda_i^* f_i(x^*) = 0$ (complémentarité).

Si $\lambda_i^* > 0$, alors $f_i(x^*) = 0$ (saturation).

Si $f_i(x^*) < 0$, alors $\lambda_i^* = 0$.

Remarque :

Les conditions du premier ordre donnent :
$\nabla_x \mathcal{L}(x^*, \lambda^*, \mathcal{V}^*) = 0 \Leftrightarrow \nabla f_0(x^*) + \sum_i \lambda_i^* \nabla f_i(x^*) + \sum_j \mathcal{V}_j^* \nabla h_j(x^*)$,
avec $\nabla f_0(x^*) + \sum_i \lambda_i^* \nabla f_i(x^*) + \sum_j \mathcal{V}_j^* \nabla h_j(x^*) = 0$.

Exemple canonique :

$A \in \mathbb{R}^{n \times p}, b \in \mathbb{R}^n, d \in \mathbb{R}^r$.

On a $\min_{x \in \mathbb{R}^p} \ \frac{1}{2} \parallel Ax - b \parallel^2$, sous la contrainte : $Cx - d = 0$ où $C \in \mathbb{R}^{rxp}$

Nous avons :

$$Cx = d \Leftrightarrow \begin{cases} C_1 : x = d_1 \\ ... \\ C_r : x = d_r \end{cases} \tag{9}$$

$\mathcal{L}(x, \mathcal{V}) = \frac{1}{2} \parallel Ax - b \parallel^2 + \mathcal{V}^T(Cx - d)$, avec $\mathcal{V}^T(Cx - d) = \sum_{i=1}^{r} \mathcal{V}_i(C_i x - d_i)$.

Est-ce que $\nabla x \ \mathcal{L} = 0$?

$\nabla x \ \mathcal{L} = A^T(Ax - b) + C^T\mathcal{V} = 0 \ \Leftrightarrow \ A^TAx^* = A^Tb - C^T\mathcal{V}^*$.

On a aussi $Cx^* = d \ \Rightarrow \ A^T(Ax^* - b) = -C^T\mathcal{V}$.

Nous avons donc le système d'équations suivant :

$$\begin{cases} A^TAx^* = A^Tb - C^T\mathcal{V}^* \\ A^T(Ax^* - b) = -C^T\mathcal{V} \end{cases} \tag{10}$$

Si x^* et $\mathcal{V}^*$ vérifient ce système, alors x^* est solution du primal.
Pour tout x fixé, nous avons :

$$\begin{aligned} \parallel Ax - b \parallel^2 &= \parallel Ax - Ax^* + Ax^* - b \parallel^2 \\ &= \parallel A(x - x^*) \parallel^2 + \parallel Ax^* - b \parallel^2 + 2\langle A(x - x^*), Ax^* - b \rangle \\ &= -2\langle x - x^*, A^T(Ax^* - b) \rangle \\ &= -2\langle x - x^*, C^T\mathcal{V} \rangle \\ &= -2\langle C(x - x^*), \mathcal{V} \rangle. \end{aligned} \tag{11}$$

Si x est faisable, $Cx = d$, et nous avons :

$$\begin{aligned} \parallel Ax - b \parallel^2 &= \parallel Ax^* - b \parallel + \parallel A(x^* - x) \parallel^2 + 0 \\ &\geqslant \parallel Ax^* - b \parallel^2 \ \ \forall x \text{ tel que } Cx = d. \end{aligned} \tag{12}$$

3 Application numérique : le problème de l'aéroglisseur

Dans cette application, il nous est exposé le problème de l'aéroglisseur : on nous donne un ensemble de k points auxquels sont associés des temps de passage, notés T. L'objectif est de passer sur les points à ces moments précis tout en minimisant le coût du déplacement pour se rendre aux différents points.

Les variables importantes pour résoudre ce problème sont :

- la position de l'aéroglisseur, notée x_t

- la vitesse, notée v_t

- le coût de déplacement de l'aéroglisseur, notée u_t.

La dynamique de l'aéroglisseur peut se modéliser de la manière suivante :

On pose $x_0 = 0$ et $v_0 = 0$.
Pour $t = 0, 1, ..., T-1$, nous avons :

$$\begin{cases} x_{t+1} = x_t + v_t \\ v_{t+1} = v_t + u_t \end{cases} \tag{13}$$

Nous avons les contraintes suivantes sur les points de passage : $x_{t_i} = w_i$ pour $i = 1, ..., k$.

Pour minimiser le coût du déplacement, nous devons minimiser : $\| u_0 \|^2 + \| u_1 \|^2 + ... + \| u_T \|^2$.

Il existe deux modèles différents pour répondre à ce problème :

- Premier modèle : on passe exactement par tous les points de passage.
Dans ce cas, nous devons minimiser $\sum_{t=0}^{T} \| u_T \|^2$.

- Deuxième modèle : nous autorisons l'aéroglisseur à passer proche des points de passage sans forcément les atteindre avec exactitude.
Dans ce cas, on doit minimiser $\sum_{t=0}^{T} \| u_T \|^2 + \lambda \sum_{i=1}^{k} \| x_{t_i} - w_i \|^2$.
Ici, λ permet de contrôler le compromis entre minimisation du coût de déplacement et passage sur tous les points.

Si nous comparons ces deux modèles, nous pouvons voir que le premier est en fait un cas particulier du second.
En effet, en partant du second modèle, nous pouvons retomber sur le premier en posant $x_{t_i} = w_i$, ce qui fait disparaître la dernière somme du modèle 2.

Pour répondre numériquement au problème de l'aéroglisseur, un code Julia nous a été fourni. Ce code permet d'illustrer les deux types de modèles existants.

Dans le cadre de notre projet, nous avons dû créer un nouveau code similaire à celui qui nous a été fourni, mais traduit en language Python.

Nous avons tout d'abord commencé par le modèle le plus simple. Dans celui-ci, nous cherchons à faire passer un aéroglisseur sur des points définis au préalable, à des temps de passage précis.

L'objectif est de minimiser le coût du déplacement de l'aéroglisseur, tout en respectant exactement les points de passages et les temps auxquels il doit s'y rendre.

Le code Python qui nous a permis de réaliser ce problème est disponible en annexe.
La trajectoire de l'aéroglisseur obtenue est la suivante :

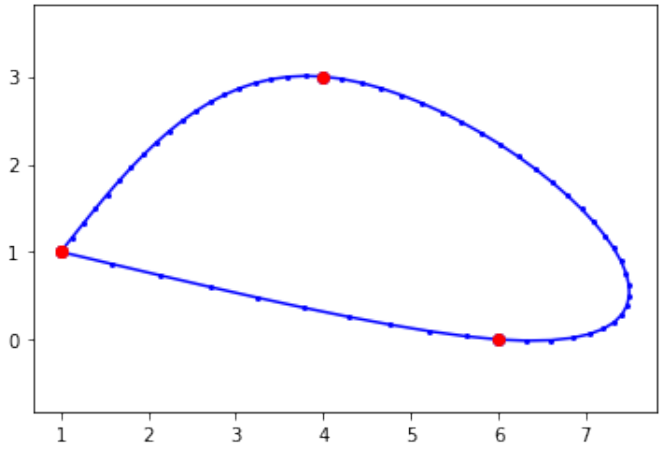

Figure 4: Optimisation convexe avec passage obligatoire sur des points prédéfinis

Nous pouvons voir que l'aéroglisseur respecte bien les points de passage obligatoires et adapte sa trajectoire selon ces points, tout en cherchant à minimiser le coût de son déplacement.

Nous allons maintenant nous intéresser à un second modèle. Cette fois-ci, nous autorisons l'aéroglisseur à manquer certains points (bien que sa trajectoire doive tout de même en rester proche).
En effet, nous rajoutons au modèle le paramètre λ (défini à l'avance), qui permet de contrôler le compromis entre minimisation du coût de déplacement et passage sur tous les points.

De la même manière que précédemment, nous fournissons le code résolvant ce problème d'optimisation en annexe, et nous exposons ci-dessous la trajectoire obtenue :

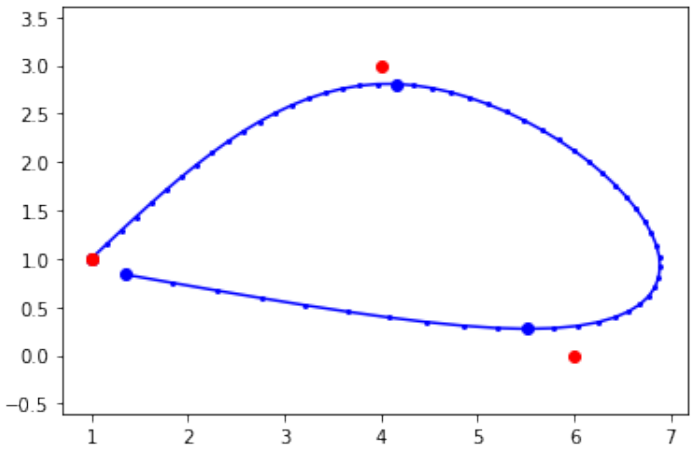

Figure 5: Optimisation convexe faisant un compromis entre minimisation du coût de déplacement et passage sur tous les points

La trajectoire obtenue minimise bien le coût de déplacement, tout en s'accordant la liberté de s'éloigner légèrement de certains points de passage.

4 Annexes :

Annexe 1 : Code du modèle 1 :

```
k = 4            # nombre de points de passage
T = np.zeros(k, int)   # vecteur des temps de passage
xw = np.zeros(k)  # vecteur des coordonnees des points de passage obligatoires

# On definit les differents temps de passage
T[0] = 1
T[1] = 20
T[2] = 50
T[3] = 60

# On definit les coordonnees des points de passage obligatoires
xw = np.array([[1, 4, 6, 1], [1, 3, 0, 1]])

# On cree les variables u (le cout du deplacement),
# x (la position de l'aeroglisseur), et v (la vitesse).
u = cp.Variable(shape=(2, T[k-1]))
x = cp.Variable(shape=(2, T[k-1]))
v = cp.Variable(shape=(2, T[k-1]))


# On definit les contraintes :

contraintes = []

for i in range(0, T[k-1]-1):
    contraintes += [
        x[:, i + 1] == x[:, i] + v[:, i],
        v[:, i + 1] == v[:, i] + u[:, i]
    ]

# xti=wi car nous sommes dans le cas ou nous devons passer exatement
# sur les points
for i in range(0, k):
    contraintes += [xw[:, i] == x[:, T[i] - 1]]

# On minimise la somme des u^2 :

objective = cp.Minimize(cp.sum(u**2))
prob = cp.Problem(objective, contraintes)
prob.solve()

# On trace le trajet passant par tous les points et minimisant
# le cout du deplacement :

py.plot(x.value[0, :], x.value[1, :], "b.-", markersize=4)
py.plot(x.value[0, :][T[0:k-1]-1], x.value[1, :][T[0:k-1]-1], "b.",
        markersize=12)
```

```
py.plot(xw[0, :], xw[1, :], "r.", markersize=12)
py.axis("equal")
```

Annexe 2 : Code du modele 2 :

```
k = 4            # nombre de points de passage
T = np.zeros(k, int)   # vecteur des temps de passage
xw = np.zeros(k)    # vecteur des coordonnees des points de passage souhaites

# On definit les differents temps de passage
T[0] = 1
T[1] = 20
T[2] = 50
T[3] = 60

# On definit les coordonnees des points de passage souhaites
xw = np.array([[1, 4, 6, 1], [1, 3, 0, 1]])

# On cree les variables u (le co t du deplacement),
# x (la position de l'aeroglisseur), et v (la vitesse).
u = cp.Variable(shape=(2, T[k-1]))
x = cp.Variable(shape=(2, T[k-1]))
v = cp.Variable(shape=(2, T[k-1]))


# On cree une fonction de parametre lambda :
def optimisation_convexe(Lambda):

    # On definit les contraintes :

    contraintes = []

    for i in range(0, T[k-1]-1):
        contraintes += [
            x[:, i+1] == x[:, i] + v[:, i],
            v[:, i+1] == v[:, i] + u[:, i]
        ]

    # On passe exactement sur le premier point de passage uniquement
    # (on s'autorise un ecart sur les autres) :
    contraintes += [xw[:, 0] == x[:, T[0]-1]]

    # On minimise la somme des u^2 + Lambda * (somme des xti - wi)^2 :

    objective = cp.Minimize(cp.sum_squares(u) +
                            Lambda*cp.sum_squares((x[:, T[0:k]-1] -
                                                   xw[:, 0:k])))
    prob = cp.Problem(objective, contraintes)
    prob.solve()

    # On trace le trajet minimisant le cout du deplacement en s'autorisant
```

```
    # quelques ecarts aux points de passages :

    py.plot(x.value[0, :], x.value[1, :], "b.-", markersize=4)
    py.plot(x.value[0, :][T[0:k]-1],
            x.value[1, :][T[0:k]-1], "b.", markersize=12)
    py.plot(xw[0, :], xw[1, :], "r.", markersize=12)
    py.axis("equal")

# On appelle la fonction optimisation_convexe et on choisit la
# valeur de lambda (ici 0.01) :
optimisation_convexe(0.01)
```