CHAPITRE 1: MÉTHODES NUMÉRIQUES POUR LE CONTRÔLE OPTIMAL

1. Introduction

Soit $T > 0$. On considère une dynamique associée à un problème de Cauchy classique, posé sur un interval $[0, T]$:

$$\begin{cases} \dot{y}(t) & = A(y(t), c(t)) \\ y(t=0) & = y_{init}. \end{cases} \tag{1}$$

Dans ce système, y_{init} est une condition initiale, A est un opérateur général, $c(t)$ est un paramètre qu'on appelle "contrôle". L'idée est que ce contrôle peut être fixé comme on veut, tel un bouton qu'on peut tourner pour ajuster une valeur.

Cadre : les points qui vont être abordés dans ce chapitre (et dans le suivant) peuvent être présentés dans un cadre de dimension finie d'espace. Autrement dit, on va considérer que $c(t) \in \mathbb{R}^n$ et $y(t) \in \mathbb{R}^m$, pour deux entiers n et m.

Dans un problème de contrôle, on se fixe généralement un état cible y_{cible} que l'on souhaite atteindre au temps T, i.e., on souhaite trouver c pour que

$$y(T) = y_{cible}.$$

Deux questions émergent alors plus ou moins naturellement :

(1) L'application $c \mapsto y(T)$ est-elle surjective ? C'est ce qu'on appelle la *contrôlabilité*.

(2) Comment calculer concrètement c ? C'est ce qu'on appelle le *contrôle optimal*.

Dans ce chapitre (et dans le suivant), nous nous intéressons à cette dernière question. De manière général le contrôle optimal consiste à transformer le problème de contrôle en un problème d'optimisation. Ceci se traduit concrètement par l'introduction d'une fonctionnelle de coût, typiquement de la forme

$$J(c) = \|y(T) - y_{cible}\|^2 + \alpha \int_0^T c^2(t) dt,$$

qui doit être minimisée sous la contrainte (1). Dans cette fonctionnelle, α est un coefficient positif qui permet d'équilibrer deux contraintes : d'une part, on souhaite résoudre le problème de contrôle, i.e., que l'état $y(T)$ soit proche de y_{cible}, d'autre part, on souhaite que le contrôle obtenu ne soit pas trop grand (en norme L^2 dans cet exemple). Dans certaines applications, les contrôles trop grands ne sont en effet pas réalisables en pratique, d'où cette deuxième contrainte. Le problème devient finalement :

$$\min_c J(c). \tag{2}$$

Remarque 1. *Le domaine du contrôle optimal est très proche du domaine des problèmes inverses, où l'enjeu est de retrouver des paramètres d'un système (par exemple des coefficients dans l'équation décrivant la dynamique) en observant une partie seulement, par exemple $y(T)$.*

Une fois dans le cadre du contrôle optimal, on peut appliquer toutes les méthodes d'optimisation numérique. Rappelons qu'une manière de classer ces méthodes consiste à les regrouper selon le nombre de dérivées utilisées : les méthodes d'ordre 0 n'utilise que la fonctionnelle (méthode de Monte-Carlo, méthode du simplex), les méthodes d'ordre 1 utilisent le gradient (méthode de descente de gradient et ses variantes), les méthodes d'ordre 2 utilisent la Hessienne (méthode de Newton et ses variantes), etc.

Remarque 2. *Rappelons le principe des méthodes de descente de gradient. Supposons que l'on dispose d'un contrôle c que l'on souhaite améliorer en le corrigeant par à l'aide d'un accroissement δc. En supposant J différentiable, nous avons :*

$$J(c+\delta c) = J(c) + \langle \nabla J(c), \delta c\rangle + o(\delta c),$$

où $\langle\cdot,\cdot\rangle$ représente le produit scalaire euclidien classiquement associé à la norme $\|\cdot\|$ pour l'espace des contrôles. On voit qu'en choisissant $\delta c = -\lambda \nabla J(c)$, avec λ positif, nous obtenons

$$J(c+\delta c) = J(c) - \lambda \|\nabla J(c)\|^2 + o(\lambda),$$

ce qui montre qu'en choisissant λ suffisamment petit, le contrôle $c - \lambda \nabla J(c)$ donne lieu à une valeur de J plus petite que celle obtenue avec c. D'où l'itération classique de la méthode du gradient à pas fixe par exemple :

$$c^{k+1} = c^k - \lambda \nabla J(c^k).$$

2. Calcul du gradient

Dans le cas du problème d'optimisation (2), le gradient ne se calcule pas directement, puisque l'expression de J contient $y(T)$ qui est elle-même une fonction de c. Nous avons donc à faire à une fonction composée.

2.1. **Deux méthodes.** Voyons deux méthodes pour calculer en pratique le gradient.

2.1.1. *Méthode directe.* Cette méthode repose sur le résultat suivant.

Lemme 1. *Soit δc, un contrôle arbitraire (considéré comme une variation). Nous avons*

$$\langle \nabla J(c), \delta c\rangle = 2(\delta y(T), y(T) - y_{cible}) + 2\alpha \int_0^T \langle \delta c(t), c(t)\rangle dt,$$

où $(\cdot,\cdot)$ représente le produit scalaire euclidien classiquement associé à la norme $\|\cdot\|$ pour l'espace des états et où $\delta y(T)$ est la valeur au temps T de la fonction définie par

$$\begin{cases} \dot{\delta y}(t) & = \partial_y A(y(t), c(t))\delta y + \partial_c A(y(t), c(t))\delta c(t). \\ \delta y(t=0) & = 0. \end{cases}$$

Ici, on a noté par ∂_y et ∂_c les différentielles de A par rapport à chacune de ses deux variables.

Nous voyons tout de suite l'inconvénient de cette approche : elle ne permet pas de calculer le gradient explicitement, mais seulement son produit scalaire avec un vecteur quelconque. Bien sûr, en petite dimension, nous pouvons alors calculer le gradient en calculant chacune de ses composantes.

2.1.2. *Méthode adjointe.* Voyons maintenant une méthode beaucoup plus puissante, puisqu'il donne directement accès au gradient. Elle repose sur l'introduction du Lagrangien, qui dans notre cas est défini par

$$\mathcal{L}(y,p,c) = J(c) - \int_0^T (p(t), \dot{y}(t) - A(y(t),c(t)))\,dt.$$

Un peu de vocabulaire : la variable p est appelé *adjoint*, *état adjoint*, ou encore *multiplicateur de Lagrange*. On appelle *système d'optimalité* ou encore *Équations d'Euler-Lagrange*, le système obtenu par annulation des dérivées partielles de $\mathcal{L}$ par rapport à ses différentes variables. Attention ! Une variable est ici précisément une quantité apparaissant dans (1) qui varie lorsque c varie. Établissons ce système ! La dérivée par rapport à $p(t)$ est facile à obtenir. Nous obtenons donc la première équation

$$\partial_{p(t)}\mathcal{L}(y,p,c) = \dot{y}(t) - A(y(t),c(t)) = 0.$$

La dérivée par rapport à y est légèrement plus compliquée à obtenir, puisqu'elle nécessite une intégration par parties. Nous avons en effet

$$\begin{aligned}\mathcal{L}(y,p,c) =& J(c) - \int_0^T (p(t), \dot{y}(t) - A(y(t),c(t)))\,dt \\ =& J(c) - (p(T),y(T)) + (p(0),y(0)) \\ &+ \int_0^T (\dot{p}(t),y(t)) + (p(t),A(y(t),c(t)))\,dt.\end{aligned}$$

Sous cette forme, nous pouvons différencier $\mathcal{L}$ par rapport à $y(t)$ et $y(T)$, qui sont les deux quantités qui varient lorsque c varie et que (1) est satisfaite. Nous obtenons les équations :

$$\partial_{y(t)}\mathcal{L}(y,p,c) = \dot{p}(t) + \partial_y A(y(t),c(t))^T p(t) = 0.$$

$$\partial_{y(T)}\mathcal{L}(y,p,c) = 2(y(T) - y_{cible}) - p(T) = 0.$$

Enfin, nous pouvons différentier $\mathcal{L}$ par rapport à $c(t)$, ce qui donne :

$$2\alpha c(t) + \partial_c A(y(t),c(t))^T p(t) = 0.$$

En résumé, le système d'optimalité est donné par :

$$\text{(3)} \qquad \begin{cases} \dot{y}(t) & = A(y(t),c(t)) \\ y(t=0) & = y_{init} \\ \dot{p}(t) + \partial_y A(y(t),c(t))^T p(t) & = 0 \\ p(t=T) & = 2(y(T) - y_{cible}) \\ 2\alpha c(t) + \partial_c A(y(t),c(t))^T p(t) & = 0. \end{cases}$$

Remarque 3. *Le système* (3) *peut être vu comme une généralisation de l'équation* $f'(x) = 0$ *que l'on considère lorsqu'on cherche les extrema d'une fonction dérivable définie sur un ouvert de* $\mathbb{R}$ *à valeurs dans* $\mathbb{R}$.

Revenons à notre motivation initiale. Les calculs précédents peuvent en fait être utilisés pour calculer le gradient de J, comme l'explique le résultat suivant.

Lemme 2. *Soit c un contrôle arbitraire, y et p les solutions de*

$$\left\{\begin{array}{cl} \dot{y}(t) & = A(y(t),c(t)) \\ y(t=0) & = y_{init} \\ \dot{p}(t)+\partial_y A(y(t),c(t))^T p(t) & = 0 \\ p(t=T) & = 2(y(T)-y_{cible}). \end{array}\right. \tag{4}$$

Alors :

$$\nabla J(c) = 2\alpha c(t) + \partial_c A(y(t),c(t))^T p(t).$$

Autrement dit, pour calculer le gradient de J en un point c, il faut résoudre les quatre premières équations de (3), et le gradient est donné par la cinquième équation.

2.2. **Introduction d'une discrétisation en temps.** La résolution numérique des équations considérées dans ce qui précède est rarement faite de manière exacte. En pratique, l'équation (1) est remplacée par une version discrétisée en temps, obtenue par une méthode numérique de résolution de problème de Cauchy. Pour traiter numériquement le problème d'optimisation, il nous faut reprendre l'ensemble des calculs précédents dans ce nouveau cadre discret.

On se donne une discrétisation de $[0,T]$ par des points $t_n = n\Delta t$, où $n = 0,\cdots,N$. La première étape consiste à définir l'équivalent discret du problème d'optimisation, c'est-à-dire des versions discrètes de la fonctionnelle J et de l'équation (1). Nous considérons ici une simple discrétisation de (1) par une méthode d'Euler explicite. En notant y_n et c_n les approximations de $y(t_n)$ et $c(t_n)$, nous sommes conduit à considérer le problème :

$$\min_c J_{\Delta t}(c), \tag{5}$$

où

$$J_{\Delta t}(c) = \|y_N - y_{cible}\|^2 + \alpha\Delta T \sum_{j=0}^{N-1} c_n^2,$$

$$\frac{y_{n+1}-y_n}{\Delta t} = A(y_n,c_n), \tag{6}$$

avec comme condition initial y_{init}. Nous pouvons alors reprendre les deux méthodes vues précédemment pour calculer le gradient.

2.2.1. *Méthode directe.* En reprenant l'approche présentée en section 2.1.1 nous obtenons

Lemme 3. *Soit δc, un contrôle arbitraire discrétisé.*

$$\langle \nabla J_{\Delta t}(c), \delta c\rangle = 2(\delta y_N, y_N - y_{cible}) + 2\alpha\Delta t \sum_{j=0}^{N-1} \langle \delta c_n, c_n\rangle,$$

où δy_N obtenue par la résolution itérative de

$$\frac{\delta y_{n+1}-\delta y_n}{\Delta t} = \partial_y A(y_n,c_n)\delta y_n + \partial_c A(y_n,c_n)\delta c_n,$$

avec la condition initiale $\delta y_0 = 0$.

2.2.2. *Méthode adjointe.* Pour appliquer la méthode adjointe, il nous faut introduire le Lagrangien discret. Posons donc

$$\mathcal{L}_{\Delta t}(y,p,c) = J_{\Delta t}(c) - \Delta t \sum_{j=0}^{N-1} \left(p_{n+1}, \frac{y_{n+1}-y_n}{\Delta t} - A(y_n,c_n) \right),$$

qu'il nous faut dériver par rapport à chacune de ses variables. Nous obtenons le système d'optimalité suivant :

$$\left\{ \begin{array}{cl} \frac{y_{n+1}-y_n}{\Delta t} & = A(y_n,c_n) \\ y_0 & = y_{init} \\ \frac{p_{n+1}-p_n}{\Delta t} & = -\partial_y A(y_j,c_n)^T p_{n+1} \\ p_N & = 2(y_N - y_{cible}) \\ 2\alpha c(t) + \partial_c A(y_n,c_n)^T p_{n+1} & = 0. \end{array} \right. \tag{7}$$

Et nous avons la du Lemme 2.

Lemme 4. *Soit c un contrôle arbitraire, y et p les solutions de*

$$\left\{ \begin{array}{cl} \frac{y_{n+1}-y_n}{\Delta t} & = A(y_n,c_n) \\ y_0 & = y_{init} \\ \frac{p_{n+1}-p_n}{\Delta t} & = -\partial_y A(y_n,c_n)^T p_{n+1} \\ p_N & = 2(y_N - y_{cible}). \end{array} \right. \tag{8}$$

Alors :

$$\nabla J_{\Delta t}(c)_n = 2\alpha c_n + \partial_c A(y_n,c_n)^T p_{n+1}.$$

Nous pouvons donc obtenir le gradient d'une manière tout à fait similaire au cas continu.

Remarque 4. *On voit qu'une fois qu'on a choisi une méthode de discrétisation pour l'équation* (1) *sur y, on a automatiquement une discrétisation sur la variable adjointe p.*

2.2.3. *Un test de debuggage.* Une fois implémentée la méthode pour calculer le gradient, il faut vérifier facilement que l'on n'a pas fait d'erreur de codage. Une méthode simple consiste à utiliser la relation classique suivante :

$$\frac{f(x+\delta x) - f(x-\delta x)}{2\delta x} - f'(x) \approx O(\delta x^2),$$

qui s'obtient en développant suffisamment les expressions $f(x+\delta x)$ et $f(x-\delta x)$. On en déduit un test pour vérifier qu'on a pas d'erreur dans le gradient. Si il n'y a pas d'erreur, un code du type :

```
c = rand(1,N-1);
dc = rand(1,N-1);
[~,gradJ]=fonctionelle(y0,c,ycible,alpha);

for epsi=10.^[-1:-1:-6]
[Jp]=fonctionelle(y0,c+epsi*dc,ycible,alpha);
[Jm]=fonctionelle(y0,c-epsi*dc,ycible,alpha);
(Jp-Jm)/(2*epsi) -gradJ*dc'
end
```

doit donner lieu à une décroissance de l'ordre de ε^2.

3. Méthode de Newton

Le problème d'optimisation peut enfin être attaqué par une méthode de Newton. Celle-ci s'écrit dans notre cas :

$$\mathrm{Hess} J_{\Delta t}(c^k) \cdot \delta c^{k+1} = -\nabla J_{\Delta t}(c^k),$$

avec

$$\delta c^{k+1} = c^{k+1} - c^k.$$

Nous voyons que δc^{k+1} est la solution d'un système linéaire. Le problème est donc de calculer $\mathrm{Hess} J_{\Delta t}(c^k)$ et même en fait $\mathrm{Hess} J_{\Delta t}(c^k) \cdot \delta c^{k+1}$.

En pratique, on peut éviter de calculer la matrice $\mathrm{Hess} J_{\Delta t}$ qui est de très grande taille. L'idée est d'utiliser un solveur de grand système ne nécessitant que la donner du code calculant le produit matrice-vecteur $v \mapsto \mathrm{Hess} J_{\Delta t}(c^k) \cdot v$. Le plus célèbre de ces solveurs, *gmres*, est implémenté dans Matlab et Octave.

Enfin, l'implémentation de la fonction se fait simplement en répétant la méthode de différenciation directe vue en Section 2.1.1. On obtient ainsi :

$$\begin{aligned}
\mathrm{Hess} J_{\Delta t}(c) \cdot \delta c =& 2\alpha \delta c(t) + [\partial^2_{c,c} A(y_n, c_n)\delta c]^T p_{n+1} \\
&+ [\partial^2_{c,y} A(y_n, c_n)\delta y]^T p_{n+1} \\
&+ \partial_c A(y_n, c_n)^T \delta p_{n+1},
\end{aligned}$$

où δy est obtenu en résolvant (3) et δp est obtenue par la résolution itérative de

$$\begin{aligned}
\frac{\delta p_{n+1} - \delta p_n}{\Delta t} =& -\partial_y A(y_n, c_n)^T \delta p_{n+1} \\
&- [\partial^2_{y,c} A(y_n, c_n)\delta c]^T p_{n+1} \\
&- [\partial^2_{y,y} A(y_n, c_n)\delta y]^T p_{n+1} \\
\delta p_N =& 2\delta y_N.
\end{aligned}$$