Résolution du problème de Commande Optimale par le principe du maximum de Pontryagin et méthode de tir

Introduction

Il existe deux grandes familles de résolution de problème de commande optimale. Les méthodes étudiées lors de ce TD/TP font partie des méthodes dîtes indirectes. Elles mettent en jeu l'approche variationnelle pour résoudre le problème de commande optimale en l'absence de contrainte. Dans le cas où des contraintes sont à considérer sur l'état et/ou la commande, le principe de Pontryagin est appélé. Il en résulte un système d'équations algrébro-différentielles dont les solutions procurent des profils d'état et de commande solutions du problème de commande optimale. Le système d'équations algébro-différentielles aussi appelé problème aux deux bouts résultant de l'écriture des conditions nécessaires d'optimalité. L'objet de la séance est d'acquérir une démarche logicielle permettant de résoudre le problème de commande optimale. La démarche proposée est une méthode dites de tir. Elle consiste en trois étapes principales :

— L'écriture du problème aux deux bouts ;
— La programmation de la fonction de tir ;
— La résolution d'un système d'équations non linéaire.

Dans un premier temps, nous aborderons le problème non contraint par une approche variationnelle.

I Problème non contraint

Considérons le système dynamique linéaire décrit par :

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} X + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{1}$$

où $X \in \mathbb{R}^2$ et $u \in \mathbb{R}$. Le problème de commande optimale doit permettre de conduire le système d'un état initial X_0 au plus près de l'état d'équilibre 0 en un temps donnée.

Le problème de commande optimale à résoudre est donnée par

$$\begin{array}{ll} \min\limits_{u} & J = \int_{t_0}^{t_f} (2x_1^2(t) + x_2^2(t) + u^2(t))dt \\ \text{sous} & \begin{cases} \dot{X} = AX(t) + Bu(t) \\ t_0 = 0, \quad X(0) = \begin{bmatrix} 2 & 1 \end{bmatrix}^T \\ t_f = 2, \quad X(tf) \text{ libre} \end{cases} \end{array} \tag{2}$$

Question 1.a Expliciter les matrices Q, R, Q_f permettant de réécrire la fonction de coût J sous la forme

$$\frac{1}{2}\int_0^{t_f} X^T(t)QX(t) + u(t)^T Ru(t)dt + \frac{1}{2}X(t_f)^T Q_f X(t_f)$$

.

Question 1.b Décrire l'Hamiltonien du problème (2) et en déduire les équations canoniques d'Hamilton. On notera la fonction état adjoint $\lambda(t)$.

Question 1.c Expliciter la commande optimale en fonction de l'état adjoint $\lambda(t)$ puis de l'état $X(t)$. On notera $P(t)$ la matrice de passage entre λ et X (matrice de Ricatti).

Question 1.d Après avoir donné l'équation différentielle que vérifie la matrice de Ricatti ainsi que sa condition aux bouts, décrire explicitement les équations différentielles que vérifient chacun des singletons de la matrice $P(t)$.

Question 1.e A l'aide de MATLAB, développer une fonction qui permette d'intégrer chaque élément de $P(t)$ de manière rétrograde ($t_f \to 0$) à partir de la condition $P(t_f) = Q_f$.

Question 1.f Construire un modèle SIMULINK mettant en oeuvre la loi de commande optimale.

Question 1.g Visualiser les trajectoires d'état et de commande. Evaluer l'amplitude maximale de commande. L'état d'équilibre est-il atteint ?
FACULTATIF : Evaluer le coût de cette commande optimale.

II 2^{nd} problème : commande à énergie minimale pour un système LTI sous contraintes

II.1 Description du problème aux deux bouts

Considérons le problème de commande optimale suivant

$$\min \int_0^2 u^2(t)dt \quad \text{sous} \begin{cases} \dot{x}(t) = -x(t) + u(t) \\ |u(t)| \leq 1 \\ x(0) = x_0 = 0 \\ x(2) = x_f = \frac{1}{2} \end{cases} \tag{3}$$

Question 2.a Ecrire l'Hamiltonien du problème 3. En déduire l'équation d'évolution de l'état adjoint.

Question 2.b Montrer, en développant le principe du maximum de Pontryagin que la commande optimale est donnée par

$$u(t) = \begin{cases} -\frac{\lambda(t)}{2} & \text{si } -2 \leq \lambda(t) \leq 2 \\ -sign(\lambda(t)) & \text{sinon} \end{cases} \tag{4}$$

Question 2.c Est-il nécessaire d'écrire les conditions de transversalité ?

Le problème aux deux bouts est un système constitué des équations d'évolution de l'état et de l'état adjoint, de la description de la commande optimale et des conditions transverses :

$$\begin{cases} \dot{x}(t) = -x(t) + u(t) \\ \dot{\lambda}(t) = \lambda(t) \\ u(t) = h(\lambda(t)) \\ x(0) = x_0 = 0 \\ x(2) = x_f = \frac{1}{2} \end{cases} \tag{5}$$

où la fonction $h(\lambda(t))$ est donnée par l'équation (4).

II.2 Mise en oeuvre logicielle du problème à valeur initiale

Afin de trouver la trajectoire qui répond aux conditions d'optimalité décrites par le problème aux deux bouts (5), la méthode de tir consiste à intégrer les trajectoires de l'état et de l'état adjoint à partir des conditions initiales. Ces trajectoires sont la solution du problème à valeur initiale :

$$\begin{cases} \dot{x}(t) = -x(t) + u(t) \\ \dot{\lambda}(t) = \lambda(t) \\ u(t) = h(\lambda(t)) \\ \text{ à partir de } x(0) = x_0 \text{ et } \lambda(0) = \lambda_0 \end{cases} \tag{6}$$

Question 2.d Quelle information manque-t-il au problème 6 pour que les trajectoires de l'état et l'état adjoint soient complètement définies ?

On note l'information manquante z.

Question 2.e Ecrire une fonction MATLAB qui prend en argument l'information manquante z et qui fournit les trajectoires $x(t)$ et $\lambda(t)$. Cette fonction utilisera les outils d'intégration numérique de MATLAB tel que `ode45`.

II.3 La Fonction de Tir

Dans le paragraphe précédent, nous avons défini une fonction MATLAB qui donne les trajectoires intégrées numériquement en fonction de z.

Question 2.f Pour $z = 1$, quelle est la valeur finale de l'état $x(2)$? Cette valeur correspond-elle à la valeur x_f donnée dans le problème (3) ?

Définissons maintenant la fonction T qui, à chaque valeur de z, associe les conditions finales sur l'état :

$$\begin{aligned} T : \quad & \mathbb{R} \to \mathbb{R} \\ & z \mapsto T(z) = x(2) - x_f \end{aligned} \tag{7}$$

où $x(t)$ est obtenu par intégration du problème à valeur intiale (6). T est appelée *fonction de tir.*

Question 2.g Ecrire une fonction MATLAB qui permet de calculer $T(z)$ en s'appuyant sur la fonction d'intégration des trajectoires obtenue à la question 2.e.

II.4 Résolution numérique du problème aux deux bouts

Résoudre le problème aux deux bouts (5) équivaut à obtenir une racine de l'équation $T(z) = 0$. Après s'être muni d'un algorithme d'intégration du système différentiel à valeur initiale pour calculer $T(z)$, l'algorithme de résolution numérique du problème aux deux bouts sera complet si l'on définit un solveur de système d'equations. L'outil MATLAB utilisé pour résoudre un tel problème est `fsolve`.

Question 2.h Trouver la valeur de z, solution de l'équation $T(z) = 0$.

Question 2.i Sur un premier graphe, tracer les trajectoires de l'état et de l'état adjoint, et, sur un second graphe, la commande optimale

Annexes

A Intégration numérique par la fonction MATLAB `ode45.m`

La fonction MATLAB `ode45.m` constitue la méthode standard pour intégrer numériquement les équations aux dérivés ordinaires de type :

$$\dot{Y}(t) = f(Y(t), t).$$

Elle est donnée par la commande suivante

$$[\texttt{Tout}, \texttt{Yout}] = \texttt{ode45}(\texttt{odefun}, \texttt{Tspan}, \texttt{Y0})$$

où

`odefun` désigne un *handle* vers la fonction décrivant la dérivée $\dot{Y}(t)$ en fonction $Y(t)$ et du temps.
- soit $f(\cdot)$ est donnée sous la forme d'une fonction MATLAB `fction_derivee` au moyen du mot clé `function`,
  `function dX=fction_derivee(t,X)`
  alors `odefun` s'écrit `@fction_derivee` ;
- soit $f(\cdot)$ est donnée par une fonction dite anonyme du type
  `fction_derivee = @(t,Y) expression_mathematique_ du_vecteur_dY/dt`,
  alors `odefun=fction_derivee`

`Tspan` designe le vecteur ligne constitué de tous les instants où l'on souhaite avoir une évaluation de Y. Au minimum, `Tspan` est constitué du temps de début et de fin de simulation : `Tspan = [t_0, t_f]` ;

`Y0` est le vecteur colonne précisant la condition initiale $Y(t_0)$;

`[Tout, Yout]` contiennent la solution numérique de l'équation aux dérivées ordinaires ($\dot{Y}(t) = f(Y(t), t)$) où la $k^{ième}$ ligne de la matrice `Yout` donne la valeur de $Y(t_k)$ où t_k est la k^{ime} valeur du vecteur `Tout` (N.B. si il a été déclarer `Tspan = [t_0, t_f]`, alors `Tout` est choisi par la fonction `ode45.m`)

`ode45.m` fait partie d'une famille de solveurs numériques d'équations aux dérivées ordinaires dont chaque membre possède ces spécificités et son domaine d'application. L'ensemble des solveurs sont inclus dans la version de base de MATLAB. Pour plus de renseignement tapez `"doc ode45"` dans MATLAB.

B Résolution numérique des systèmes d'équations non linéaires par `fsolve.m`

La fonction `fsolve` est incluse dans la boite à outils additionnelle OPTIMIZATION TOOLBOX de MATLAB. Cette fonction permet de résoudre le problème suivant

$$F(x) = 0$$

où $x \in \mathbb{R}^n$ et $F \in \mathbb{R}^m$. l'appel de `fsolve` est donnée par la commande suivante :

$$\texttt{x} = \texttt{fsolve}(\texttt{fun}, \texttt{x0})$$

où

`x` est l'évaluation numérique d'une racine du système $F(x) = 0$;

`fun` désigne un *handle* vers la fonction décrivant la fonction $F(x)$:

- soit $F(\cdot)$ est codée sous la forme d'une fonction MATLAB `systeme_nl` au moyen du mot-clé `function` à la suite de l'entête suivant :
  `function Fx=systeme_nl(X)`
  alors `fun` s'écrit `@systeme_nl` ;
- soit $F(\cdot)$ est donnée par une fonction dite anonyme du type
  `systeme_nl = @(t,Y) expression_mathematique_ du_vecteur_F(x)`,
  alors `fun=systeme_nl` ;

`x0` est une solution qui sera le point de départ de l'algorithme de résolution.

Pour plus de détail sur les algorithmes utilisés et les différentes options, tapez dans l'invite de commande MATLAB :

`doc fsolve`

C Transmettre des données supplémentaires aux fonctions *handle* par la techniques des fonctions imbriquées

Lors de la résolution des problèmes proposés, il est nécessaire de fournir aux fonctions *handle* (`@fction_derivee` et `systeme_nl`) plus de détails que les arguments de base. Il y existe plusieurs méthodes pour le faire. Une de ces méthodes consiste à écrire dans un fichier une fonction qui contiendra l'appel à la fonction `ode45` et `fsolve` mais aussi les fonctions *handle* (`@fction_derivee` et `systeme_nl`).

```
function [Tout,Yout] =  integration(a,b,c,Y0)
[Tout,Yout] = ode45(@fction_derivee,Tspan,Y0);
...
% Suite du code
...
% Inclure a la fin de la fonction integration la fonction handle
    function dydt = fction_derivee(t,Y)
            dydt=... % code du calcul de la derivee de Y utilisé a, b, c
    end % ATTENTION : finir les fonctions annexes par le mot-clef "end"
...
% Possibilite de mettre d'autres fonctions annexes
...
end% ATTENTION : finir la fonction principal par le mot-clef "end"
```

En procédant de cette façon, la fonction `@fction_derivee` a accès aux paramètres `a, b, c`. Finalement, pour obtenir de les trajectoires il suffit de lancer les commandes suivantes :

```
a = 4; b = 2.1; c = 4;% Declarer les parametres ...
y0 = [0.5,0.5]; % et les conditions initiales
[tps,sortie] = integration(a,b,c,y0);
```

Il en va de même pour l'utilisation de la fonction `fsolve` et de sa fonction *handle* `systeme_nl`.