Résolution du problème de Commande Optimale par méthode directe et collocation

1 Introduction

L'objet de ce TP est de découvrir les possibilités des méthodes directes de résolution du problème de commande optimale. Contrairement aux méthodes indirectes dont la stratégie consiste à résoudre un système d'équations algébro-différentielles, $F(Z) = 0$ obtenu par analyse des conditions d'optimalité, les méthodes directes s'attachent à minimiser directement la fonction de coût du problème de commande optimale. Les méthodes directes utilisent les algorithmes de programmation mathématique (d'optimisation) pour résoudre le problème de commande optimale. Néanmoins, les outils de programmation mathématique ne sont pas capables de prendre en compte des contraintes différentielles telles que la satisfaction de la dynamique du système ($\dot{x} = f(x, u)$). Ainsi, afin de convertir le problème d'optimisation dynamique (problème de commande optimale) en un problème de programmation mathématique, il est nécessaire de paramétrer les profils temporels de la commande et/ou de l'état sur une base prédéfinie. Les paramètres de ces différentes trajectoires forment alors le vecteur des variables de décision (ou variables d'optimisation). Ces paramètres doivent est calculer de manière à ce que la dynamique du système soit le plus précisément respectée. Il existe plusieurs familles de méthodes directes, nous citons ici les plus classiques :

Les méthodes de tir Seules les entrées de commandes sont paramétrées et considérées comme les variables de décision. Les trajectoires de l'état sont intégrées explicitement.

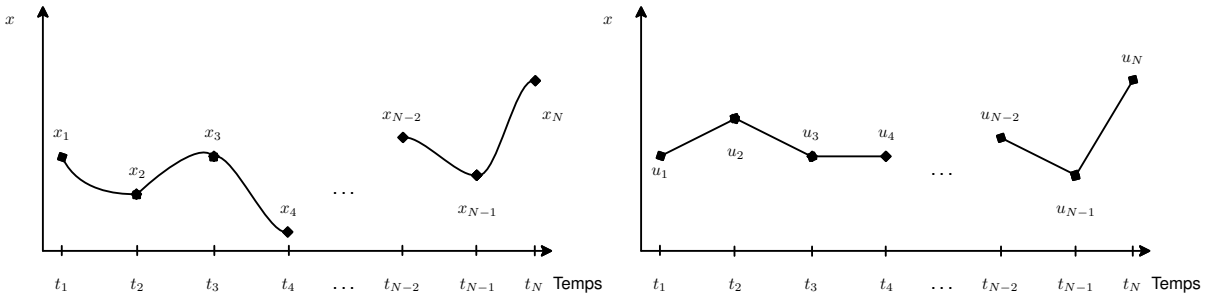
Les méthodes de transcription L'état et la commande sont paramétrés et considérés comme les variables de décision. La satisfaction de la dynamique est prise en charge par des techniques d'intégration implicite.

Le grand intérêt des méthodes directes de commande optimale réside dans la possibilité de prendre en compte un nombre important de contraintes sur la trajectoire contrairement aux méthodes indirectes pour lesquelles des contraintes, même simples, induisent une complexité accrue dans les conditions d'optimalité (présence d'arcs singuliers, difficultés à établir des structures de commutation de la commande, etc.). La contrepartie est que la solution obtenue par des méthodes directes est beaucoup moins précise du point de vue de l'optimalité (ex : consommation) que tous les exemples où les méthodes indirectes proposent une solution qui puisse être mise en oeuvre.

2 Méthode directe de transcription : Collocation

2.1 Conversion du problème de commande optimale en programme mathématique

La méthode développée dans ce TP est une méthode directe de transcription par collocation. **L'état et la commande sont discrétisés** sur un horizon de temps déterminé et la dynamique est vérifiée par



(a) Approximation cubique par morceaux de l'état (b) Approximation linéaire par morceaux de la commande

FIGURE 1 – Schéma de discrétisation

la collocation de celle-ci et d'une approximation en un nombre fini de points. Considérons le problème suivant de commande optimale sous contraintes :

$$\min_{x,u} J(x,u) = \int_{t_0}^{t_f} L(t)dt + \phi(tf)$$
$$\text{sous} \begin{cases} \dot{x}(t) = f(t, x(t), u(t)) & \text{(dynamique du système)} \\ \psi_0(t_0, x(t_0)) = 0 & \text{(condition initiale)} \\ \psi_f(t_f, x(t_f)) = 0 & \text{(condition finale)} \\ \Gamma(t, x(t), u(t)) \leq 0 & \text{(contraintes sur la trajectoire)} \end{cases} \tag{1}$$

La conversion du problème de commande optimale (1) en problème de programmation mathématique commence avec la définition d'un grille temporelle de discrétisation $t_k = \{t_1, t_2, \ldots, t_N\}$ telle que

$$t_0 = t_1 \leq t_2 \leq \cdots \leq t_{N-1} \leq t_N = t_f \tag{2}$$

Nous nommerons les instants t_i noeuds de discrétisation. Leur nombre N et leur répartition sont toujours laissés à la discrétion de l'utilisateur. En pratique, N est choisi de manière à respecter un compromis entre la précision du résultat vis-à-vis de la satisfaction des contraintes et les dimensions du problème d'optimisation. Quant à la distribution des noeuds, elle est, de manière classique, equirépartie i.e. les distances entre deux instants successifs de discrétisation sont égales. La seconde étape consiste à remplacer les fonctions $x(t)$ et $u(t)$ par leurs valeurs aux noeuds à savoir les séquences :

$$x_k = \{x_1, x_2, \ldots, x_{N-1}, x_N\} \tag{3}$$
$$u_k = \{u_1, u_2, \ldots, u_{N-1}, u_N\} \tag{4}$$

La trajectoire est ensuite reconstituée entre les instants de discrétisation grâce à des interpolations polynomiales en générale. Dans la méthode présentée ci-après la trajectoire d'état sera décrite par un polynôme de degré 3 sur chaque segment et la commande est linéaire par morceau (cf les figures 1).

Dans la méthode directe présentée, les variables de décision Ψ sont constituées de x_k et u_k.

Le principe de paramétrisation des trajectoires de l'état et de la commande permet de traduire la fonction de coût :

$$J(x,u) \equiv F(Z) = F(x_k, u_k)$$

Ainsi la solution du problème de commande optimale (1) est approchée par la solution du programme mathématique suivant

$$\min_{Z} F(Z)$$
$$\text{sous } \begin{cases} C(Z, t_k) = 0 \\ \Gamma_D(Z, t_k) \leq 0 \end{cases}$$

Les contraintes d'égalité $C(\cdot)$ doivent permettre de satisfaire la dynamique du système, et les conditions initiales et finales. La satisfaction de la dynamique est réalisée au travers de schéma d'intégration numérique spécifique.

Si le principal intérêt de cette conversion est de permettre une prise en compte aisée de contraintes de différentes natures et complexités (telle que un état initial libre, des contraintes intégrales, des contraintes de passage, etc), **la contrepartie est qu'aucune garantie de satisfaction des contraintes n'est offerte entre les points de vérification.**

2.2 Intégration numérique par collocation : méthode de Hermite-Simpson

Il existe plusieurs familles de schémas d'intégration : les schémas explicites et les schémas implicites Les méthodes de collocation impliquent l'utilisation de schémas implicites. Toutes les méthodes de collocation sont basées sur le remplacement de l'intégration de la dynamique par la réduction de résidus sur chaque segment de temps (distance entre deux noeuds consécutifs). Ces résidus, qui s'appuient sur les valeurs de x_k et u_k, doivent être égalés à zéro afin que la dynamique soit assurée.

Le schéma de collocation directe de Hermite-Simpson repose sur la paramétrisation des trajectoires par des polynômes cubique dits de Hermite sur chaque segment de temps. En effet, l'intérêt d'un polynôme cubique est qu'il possède 4 coefficients qui sont déterminés de manière unique en imposant aux extrémités du segment la valeur du polynôme et de sa dérivée. Or, nous possédons ces informations sur chaque segment k par la connaissance des variables de décision que sont l'état (x_k et x_{k+1}) et la commande (u_k et u_{k+1}). La dérivée aux extrémités est obtenue par simple calcul de la dynamique $f(x_k, u_k)$ et $f(x_{k+1}, u_{k+1})$.

Dans le cadre de la méthode de Hermite-Simpson, le résidu consiste en la différence entre l'évolution de l'état x sur un intervalle et une estimation de cette évolution obtenue par quadrature. Cette quadrature implique le calcul de la dynamique $f(x, u)$ aux instants de collocation τ_i qui sont au centre des segments $\tau_i = \frac{t_i + t_{i+1}}{2}$. (cf. figure 2).

D'une part, la trajectoire de l'état entre deux noeuds étant approchée par une fonction cubique, il est facile de calculer une estimation de l'état au milieu du segment x_i^c :

$$x_i^c = \frac{1}{2}(x_i + x_{i+1}) - \frac{t_{i+1} - t_i}{8}(f(x_{i+1}, u_{i+1}) - f(x_i, u_i))$$

D'autre part, il est aussi possible de calculer la dynamique au centre du segment :

$$f_i^c = f(x_i^c, u_i^c)$$

en considérant que la commande est linéaire par morceaux $u_i^c = \frac{1}{2}(u_i + u_{i+1})$

De cette manière, nous obtenons l'expression des résidus (5) pour chaque intervalle :

$$R_i = x_{i+1} - x_i - \frac{t_{i+1} - t_i}{6}(f_i + 4f_i^c + f_{i+1}), \quad i = 1, \ldots, N-1$$

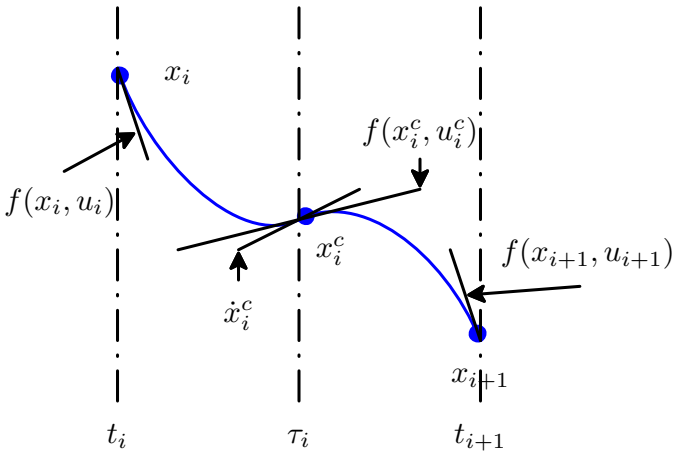


FIGURE 2 – Collocation par les polynômes d'Hermite

où $f_i = f(x_i, u_i)$.

Il vient alors que la satisfaction de la dynamique consiste en l'ensemble de contraintes égalitaires suivantes :

$$R_i = x_{i+1} - x_i - \frac{t_{i+1} - t_i}{6}(f_i + 4f_i^c + f_{i+1}) = 0, \quad i = 1, \ldots, N-1 \quad (5)$$

2.3 Evaluation des contraintes sur la grille de discrétisation

Les contraintes sur la trajectoire sont discrétisées à l'instar de la contrainte de dynamique. La principale raison en est que la trajectoire de l'état n'est réellement connue qu'aux instants de discrétisation t_k. Or, les contraintes sur la trajectoire sont représentées dans le problème de commande optimale (1) par la fonction "contrainte" $\Gamma(\cdot)$ telle que

$$\begin{array}{rcl} \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m & \rightarrow & \mathbb{R}^l \\ t, x(t), u(t) & \mapsto & \Gamma(t, x(t), u(t)) \end{array}$$

Lorsque la fonction contrainte $\Gamma(\cdot)$ est évaluée sur la grille d'instants t_k, on obtient la fonction $\Gamma_D(\cdot)$ donnée par

$$\begin{array}{rcl} \mathbb{R}^N \times \mathbb{R}^{n \times N} \times \mathbb{R}^{m \times N} & \rightarrow & \mathbb{R}^{l \times N} \\ t_k, x_k, u_k & \mapsto & \Gamma_D(t_k, x_k, u_k) = \begin{bmatrix} \Gamma(t_1, x_1, u_1) \\ \Gamma(t_2, x_2, u_2) \\ \vdots \\ \Gamma(t_N, x_N, u_N) \end{bmatrix} \end{array}$$

Ainsi, la contrainte $\Gamma(t, x(t), u(t)) \leq 0$ est remplacée par la contrainte inégalitaire suivante :

$$\Gamma_D(t_k, x_k, u_k) = \begin{bmatrix} \Gamma(t_1, x_1, u_1) \\ \Gamma(t_2, x_2, u_2) \\ \vdots \\ \Gamma(t_N, x_N, u_N) \end{bmatrix} \leq 0 \tag{6}$$

2.4 Résolution numérique du problème de commande optimale discrétisé

En appliquant au problème de commande optimale (1) les principes d'intégration numérique par collocation et en discrétisant les contraintes sur la trajectoire, nous obtenons un problème d'optimisation paramétrique non linéaire sous contraintes :

$$\min_Z F(Z)$$
$$\text{sous} \begin{cases} \begin{bmatrix} R_1(t_1, x_1, u_1, t_2, x_2, u_2) \\ R_1(t_2, x_2, u_2, t_3, x_3, u_3) \\ \vdots \\ R_1(t_{N-1}, x_{N-1}, u_{N-1}, t_N, x_N, u_N) \end{bmatrix} = 0 & \text{(contraintes de collocation)} \\ \psi_0(t_0, x(t_0)) = 0 & \text{(condition initiale)} \\ \psi_f(t_f, x(t_f)) = 0 & \text{(condition finale)} \\ \begin{bmatrix} \Gamma(t_1, x_1, u_1) \\ \Gamma(t_2, x_2, u_2) \\ \vdots \\ \Gamma(t_N, x_N, u_N) \end{bmatrix} \leq 0 & \text{(contraintes sur la trajectoire)} \end{cases}$$

MATLAB possède un algorithme d'optimisation paramétrique non linéaire sous contraintes au sein de la boite à outils OPTIMIZATION TOOLBOX : `fmincon`. Cette fonction a pour objet de calculer le minimum du problème d'optimisation suivant :

$$\min_z f(z)$$
$$\text{sous} \begin{cases} c(z) \leq 0 \\ c_{eq}(z) = 0 \\ Az \leq B \\ A_{eq} z = B_{eq} \\ z_{min} \leq z \leq z_{max} \end{cases} \tag{7}$$

Soient les variables de décision z, $c(z)$ et $c_{eq}(z)$ des fonctions non linéaires vectorielles. Les matrices A et A_{eq} et les vecteurs B et B_{eq} permettent de définir des contraintes linéaires. Enfin z_{min} et z_{max} sont les bornes sur les variables de décision z. La syntaxe est la suivante :

$$[\texttt{zopt}, \texttt{fval}, \texttt{exitflag}, \texttt{output}] = \texttt{fmincon}(\texttt{fun}, \texttt{z0}, \texttt{A}, \texttt{b}, \texttt{Aeq}, \texttt{beq}, \texttt{lb}, \texttt{ub}, \texttt{nonlcon}, \texttt{options})$$

où les entrées de la fonction `fmincon` sont

`fun` désigne un *handle* vers la fonction décrivant la fonction objective $f(z)$

– soit $f(z)$ est donnée sous la forme d'une fonction MATLAB `fction_cout` au moyen du mot clé `function`,
`function dX=fction_cout(z)`
alors `fun` s'écrit `@fction_cout` ;
– soit $f(\cdot)$ est donnée par une fonction dite anonyme du type
`fction_cout = @(z) expression_mathematique_du_cout`,
alors `fun=fction_cout`

`z0` est la solution initiale, z_0,nécessaire à la résolution ;

`nonlcon` désigne un *handle* vers la fonction fournissant à la fois l'évaluation des contraintes non linéaires $c(\cdot)$ et $c_eq(\cdot)$ pour une valeur donnée de z. `nonlcon` s'écrit `@contraintes_nonlin` où `contraintes_nonlin` est une fonction MATLAB fournissant deux sorties telle que

```
function [C,Ceq]=contraintes_nonlin(z)
C=... ; % Calcul des contraintes inegalites c(z)
Ceq=... ; % Calcul des contraintes egalites ceq(z)
end
```

`A, b, Aeq, beq, lb, ub` permettent de définir les contraintes de type linéaire et des bornes sur les valeurs de x. Si le problème ne possède pas de contraintes linéaires égalitaires, alors `Aeq=[]` et `Beq=[]`. En l'absence de contraintes linéaires inégalitaires alors `A=[]` et `B=[]`. S'il n'existe pas de bornes explicies sur les variables de décision, `lb=[]` et `ub=[]`.

`options` est une structure MATLAB regroupant l'ensemble des options de résolution du problème. Pour plus d'informations sur l'ensemble des options, il faut se réferer à la fonction `optimset` en tapant `"doc optimset"`

Les principales sorties de `fmincon` sont

`xopt` donne la valeur optimale obtenue par le solveur

`fval` fournit la valeur de la fonction de coût pour `xopt`.

`exitflag` est un entier signé qui décrit le statut du solveur lorsqu'il arrête son calcul. Si cette valeur est 1 alors l'optimisation a été menée avec succès. Pour les autres valeurs, il convient de se référer à la documentation (`"doc fmincon"`).

`output` est une structure MATLAB contenant des informations sur le processus d'optimisation telles que l'algorithme utilisé et le nombre d'itération par exemple. Plus de détails sont présents dans la documentation (`"doc fmincon"`).

La fonction `fmincon` est un outil d'optimisation globale au sens où il traite de problèmes non linéaires pouvant être complexes. Cependant il utilise des techniques d'optimisation locale. Ainsi, si les contraintes du problème sont non convexes (dans la plupart des cas), il se peut que la solution obtenue soit uniquement un minimum local dépendant de la solution initiale fournie par l'utilisateur.

3 Problème d'optimisation de direction de poussée sous contraintes

Le problème à résoudre est, pour une grande partie, identique au problème d'optimisation de direction de poussée résolu dans le TP "Résolution du problème de Commande Optimale par le principe du maximum de Pontryagin et méthode de tir". Toutefois, ici, la trajectoire est astreinte à un domaine de vol particulier pour des raisons de sécurité. De plus, la direction de poussée est contrainte à ne

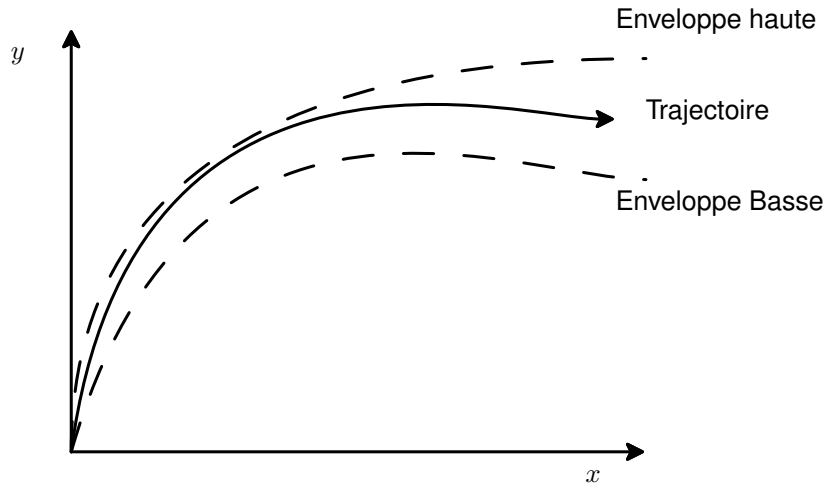


FIGURE 3 – Schéma du problème de direction de poussée

pas trop varier pour ne pas casser l'engin. Nous rappelons, ici, le modèle et les données du problème. Le véhicule considéré est mû par une poussée de norme constante a. La commande de ce véhicule est représentée par θ, l'orientation du vecteur de poussée dans le plan vertical de la trajectoire. On considère qu'au départ de la mission, le véhicule est à l'arrêt au centre du repère. On cherche à maximiser la vitesse horizontale, à l'instant final de la mission alors qu'il atteint une altitude fixée et une vitesse verticale nulle.

Le modèle de la dynamique du véhicule est donné par les équations suivantes :

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = f(X, \theta) = \begin{bmatrix} v_x \\ v_y \\ a\cos\theta \\ a\sin\theta \end{bmatrix} \tag{8}$$

Les conditions initiales sont données par

$$x(0) = y(0) = v_x(0) = v_y(0) = 0$$

et les conditions finales par

$$y(t_f) = y_f, \quad v_y(t_f) = 0$$

Le problème de commande optimale s'écrit alors :

$$\min -v_x(t_f)$$
$$\text{sous} \begin{cases} \dot{X} = f(X, \theta) \\ x(0) = y(0) = v_x(0) = v_y(0) = 0 \\ x(t_f) \text{ libre}, \ y(t_f) = y_f, \ v_y(t_f) = 0 \\ -\bar{\vartheta} \leq \frac{d\theta}{dt} \leq \bar{\vartheta} \quad \text{(Contrainte sur la commande)} \\ \gamma_b(x(t), y(t)) \leq 0 \text{ (Enveloppe basse)} \\ \gamma_h(x(t), y(t)) \leq 0 \text{ (Enveloppe haute)} \end{cases} \tag{9}$$

Le problème est normalisé : la norme de poussée a et la durée de la mission t_f sont données par

$$a = 1, \quad t_f = 1, \quad y_f = 0.2., \quad \bar{\vartheta} = 0.65$$

Les contraintes d'enveloppes sont données par :

$$\gamma_b(x,y) = 14,0625x^3 - 12.1875x^2 + 3x - y, \quad \text{(Enveloppe basse)} \tag{10}$$
$$\gamma_h(x,y) = y - 17.1875x^3 + 15.3125x^2 - 4x, \quad \text{(Enveloppe haute)} \tag{11}$$
$$\tag{12}$$

Afin de résoudre le problème (9) par la méthode de collocation décrite précédemment, nous mettrons en place l'algorithme de résolution en s'appuyant sur la structure de programme donné ci-après

```
function main
%% Declaration des donnees du pb
a=1; tf=1;
yf=0.2; % condition finale
Vtps=...; % Grille de discretisation

%% Definition des contraintes lineaires
A=...; B=...; % matrices des contraintes inegalites
Aeq=...; Beq=...; % matrices des contraintes egalites
lb=...; ul=...; % bornes sur les variables d'optimisation

%% Declaration des donnes d'optimisation
Z0=...; % Solution initiale la programmation non lineaire

%% STRUCTURATION specifique du vecteur:
%  Z : [x1...xN,y1...yN,vx1...vxN,vy1...vyN,u1...uN]

options=optimset(...);
%% Appel du solveur d'optimisation
[Zopt,fval,exitflag,output] =...
       fmincon(@fction_cout,Z0,A,b,Aeq,beq,lb,ub,@contraintes_nonlin,options);

%% fonction cout
function cout=fction_cout(Z)
% structuration specifique du vecteur Z
cout=...; % Calcul du cout
end

%% fonction de calcul des contraintes
function [C,Ceq]=contraintes_nonlin(Z)
% structuration specifique du vecteur Z
C=... ; % Calcul des contraintes inegalites C(Z)<=0
Ceq=... ; % Calcul des contraintes egalites Ceq(Z)=0
end

end
```

3.1 Collocation vs Intégration

On souhaite en premier lieu vérifier que la méthode de collocation permet effectivement de générer des trajectoires qui respectent la dynamique. D'une part, l'utilisation de schéma implicite de collocation implique l'utilisation d'algorithme et fonctions du type `fsolve.m` ou `fmincon.m` pouvant résoudre un système d'équations non linéaires. D'autre part, un problème aux conditions initiales, $\dot{x} = f(x, u)$ avec $x(t_0) = x_0$, peut être résolu numériquement par des schéma de résolution explicite de type `ode45.m`.

Question 3.a Ecrire une fonction MATLAB qui permette de calculer les résidus R_k de la dynamique donnée dans l'équation (8) en fonction des valeurs discrétisées de l'état et la commande, x_k et u_k et de la grille de discrétisation.

`function Ri = calculresidus(Z)`

Il est à noter que `Ri` est un vecteur et que le vecteur Z doit satisfaire une structuration qui sera conservée tout au long de l'exercice.

Question 3.b Modéliser sous la forme de contrainte matricielle linéaire en fonction du vecteur Z les conditions permettant de fixer la commande discrétisée $u_k = 1$pour tout k et imposer $x_1 = [0,\ 0,\ 0,\ 0]^T$. Cette modélisation dépend fortement de la structuration choisie de Z.

Question 3.c Calculer une trajectoire discrétisée particulière x_k grâce à la fonction `fmincon.m`. Pour ce faire, vous utiliserez la fonction précédemment réalisée, vous fixerez la commande discrétisée $u_k = 1$pour tout k et vous imposerez $x_1 = [0,\ 0,\ 0,\ 0]^T$.

Question 3.d Comparer la trajectoire x_k avec le résultat d'une intégration explicite

3.2 Problème aux deux bouts

Question 3.e Modéliser sous la forme de contrainte matricielle linéaire en fonction du vecteur Z les conditions initiale et finale.

Question 3.f Définir le coût en fonction des variables de décision x_k et u_k et de la grille de discrétisation. Programmer la fonction `fction_cout`

Question 3.g Completer le squelette de fonction MATLAB proposé plus haut afin de calculer une trajectoire optimale satisfaisant la dynamique et le conditions aux deux bouts.

3.3 Prise en comptes des contraintes sur la trajectoire et la commande

Le problème résolu dans la section précédente n'est pas contraint. Il s'agit maintenant d'inclure au précédent problème les spécifications sur la trajectoire et l'entrée de commande.

Question 3.h Comment la contrainte sur la dérivée de la commande peut-elle être discrétisée ? Modéliser cette contrainte sous forme d'une contrainte linéaire, inégalitaire et matricielle.

Question 3.i Ecrire une fonction MATLAB qui fournit l'évaluation des contraintes d'enveloppe aux instants de discrétisation. Les sorties de cette fonction seront deux vecteurs :

`function[Cont_envel_haute,Cont_envel_basse]=calcul_contraintes(Z)`

Puis compléter la fonction `contraintes_nonlin` à l'aide de la fonction prédedemment écrite et de la fonction `calculresidus`.

Question 3.j Mettre en place l'algorithme de génération de trajectoire optimale en adaptant le squelette de fonction MATLAB proposé plus haut.

Question 3.k Analyser la trajectoire complète de chaque état et le profil de la commande obtenu. Vérfier graphiquement que toutes les contraintes sont satisfaites aux points de collocation. Identifier les instants ou les contraintes peuvent ne pas l'être et proposer une solution pour remédier au problème.

Rappelons que la commande linéaire par morceaux et la trajectoire de l'état sont un polynôme de degré 3 sur chaque segment. Il sera ainsi judicieux d'utiliser les commandes MATLAB `interp1` et `polyval`