TP2-Commande avec anticipation, commande prédictive

Icare SAKR & Yan CHEN

Décembre 2020

1 Résumé du cours

Le modèle dynamique permet de prendre en compte des forces et, par intégration, prédire des futurs d'un système (prennant en compte des contraintes) et ainsi anticiper des séquences de commandes minimisant un coût. Le modèle le plus courant est le NMPC (non linéaire). L'horizon de ces prédictions pourrait être approchée par essais. Plusieurs algorithmes (linéarisation, discrétisation) permettent de simplifier, stabiliser et optimiser ces problèmes de commande.

2 Introduction

Dans ce TP, on vois l'effet d'une commande anticipative et une commande prédictive. Le TP est composé de trois exercices. Le premier montre l'influence de l'anticipation sur la performance du contrôle. Ensuite,l'exercice 2 nous permet de vérifier la zone de stabilité pour une commande prédictive et, finalement, l' exercice 3 pour écrire et essayer la commande prédictive. Ces deux dernières questions suivent l'article [1] en utilisent le même modèle et la théorie développés dans l'article.

3 Anticipation

Dans cette section, on voit l'effet de l'anticipation sur une trajectoire. L'idée d'anticipation est ce que le robot anticipe les positions des points dans une trajectoire selon taille de horizon. Puis il arrive position de goal par contrôle optimal(PIP). Dans le TP précédent, on utilisait seulement le PID pour contrôler l'état courant et permettre au le robot de suivre une trajectoire sans anticipation.

Code

```
while size(list_points,2) < window_size
    %% sliding of horizon if close but add point for next goal
```

```
    error = Path(:,goalWaypointId) - xtemp;
    if norm(error(1:2)) < rho
        xtemp=Path(:,goalWaypointId);
        list_points = [list_points,xtemp];
        goalWaypointId=goalWaypointId+1;
        goalWaypointId = min(goalWaypointId,size(Path,2));
    else
        direction=Path(:,goalWaypointId) - xtemp;
        direction=direction/norm(direction);
        xtemp=xtemp+dmax*direction;
        list_points=[list_points,xtemp]; %next point in direction of path end
    end
end
```

$K_\rho = 10$ et $K_\alpha = 5$ sont utilisés dans ce TP et TP précédent. Figure 2 présent que le robot peut suivre la trajectoire de référence bien avec PID. Il y a une grosse erreur au coin (près des waypoints au milieu). Puis il peut se remet rapidement sur la trajectoire de référence. L'erreur total est 361.2. Cependant, le robot contrôlé par MPC met plus de temps à se remettre sur la trajectoire de référence après de changement de path dans le Figure 1. Le robot avec anticipation (Horizon = 5) a une plus grande erreur totale 865.93. Mais il a une réponse plus rapide et anticipe que son but est d'atteindre le waypoint suivant et non de suivre nécessairement la trajectoire, ainsi nous avons uun contrôle plus souple sans changements brusques de direction et perte de temps en suivant la trajectoire si le but est d'atteindre les waypoints (différence entre Figure 1 et Figure 2).

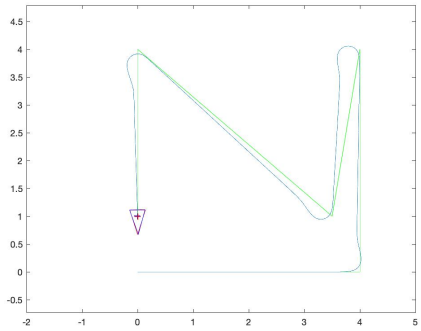


FIGURE 1 – Horizon=5

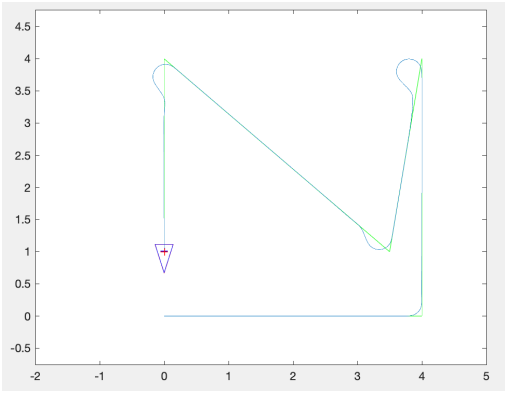


FIGURE 2 – Non horizon

Le temps de computation est plus grand plus l'horizon d'anticipation croit (temps de calcul de toutes les anticipations). Le robot peut anticiper plus de points sur la trajectoire pour un plus grand horizon. En d'autre termes, la capacité d'anticipation est plus forte. Lorsque le chemin change beaucoup, le robot peut anticiper la situation, puis trouve une meilleure chemin (plus court et plus souple) pour le suivre. La capacité est présentés évidement lorsque horizon = 100 et 1000 sur Figure 5 et Figure 6.

Horizon	Erreur totale
1	867.62
5	865.93
20	1014.93
100	2405.07
1000	1687.98

TABLE 1 – Erreur en fonction de la taille de l'horizon d'anticipation

Le meilleur horizon d'anticipation est de 5, et c'est plus bien que pour une fenêtre de taille 1 car dans ce cas on n'anticipe pas bien et on a des dépassements au niveau des virages (c'est visible pour le 1er virage (différence entre figure 3 et 1)).

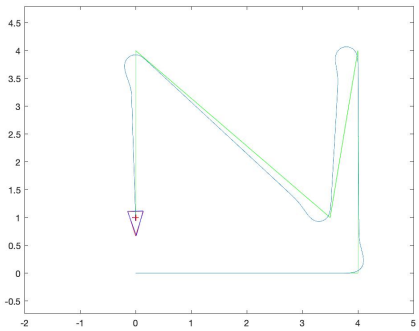


FIGURE 3 – Horizon=1

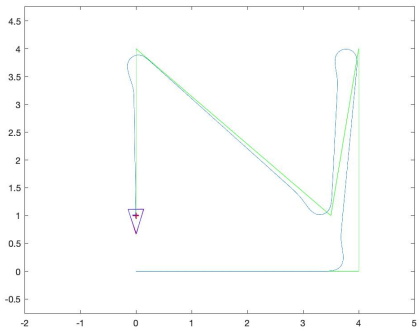


FIGURE 4 – Horizon=20

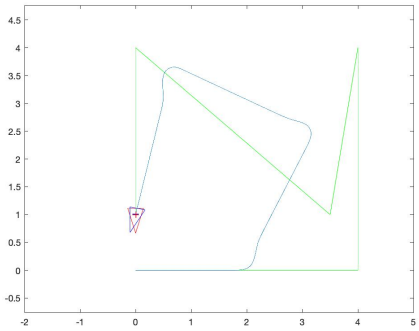


FIGURE 5 – Horizon=100

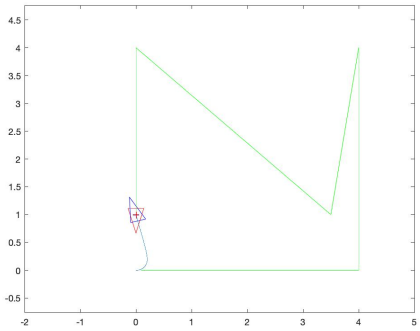


FIGURE 6 – Horizon=1000

4 Zone de stabilité d'une commande prédictive

Dans cette section, on voit la méthode de jugement de stabilité **Chen&Allgower 1998** [1] pour vérifier si un point est dans la zone de stabilité d'un contrôleur en

visualisation de stabilité. Le système étudié est le suivant :

$$\begin{aligned} \dot{x_1} &= x_2 + u(\mu + (1-\mu)x_1) \\ \dot{x_2} &= x_1 + u(\mu - 4(1-\mu)x_2) \end{aligned} \tag{1}$$

Nous choisissons $\mu = 0.5$. Le contrôle est borné $u \in [-2, 2]$.

On calcule les matrice A et B de la linéarisons Jacobienne avec les équations suivant :

$$\begin{aligned} A = \frac{\partial f}{\partial x}(0,0) &= \frac{\partial \dot{x}}{\partial x}(0,0) \\ &= \begin{bmatrix} u_0(1-\mu) & 1 \\ 1 & -4u_0(1-\mu) \end{bmatrix} \end{aligned} \tag{2}$$

$$\begin{aligned} B = \frac{\partial f}{\partial u}(0,0) &= \frac{\partial \dot{x}}{\partial u}(0,0) \\ &= [\mu + (1-\mu)x_1(0), \mu - 4(1-\mu)x_2(0)] \end{aligned} \tag{3}$$

L'utilisation de la fonction care de Matlab permet de retrouver la solution de l'équation de Riccati (c.f. code), et ainsi trouver une relation localement stabilisante $u = K \times x$ pour le contrôle par retour d'état.

L'équation du système fermé est donnée par : $A_k = A + BK$

En claculant la plus grande valeur propre de A_k nous obtenons effectivement celle de l'article : $\lambda_{max}(A_k) = -1.0$.

A partir de cette valeur propre nous résolvons l'équation de Lyapunov pour obtenir comme solution une matrice P nous servant à déterminer la région de stabilité au sens de Chen & Allgower, que nous avons visualisé sur la figure 8. Elle est de forme elliptique. [1].

On obtient la même matrice du régulateur que [1] $K = \begin{bmatrix} 2.118 & 2.118 \end{bmatrix}$

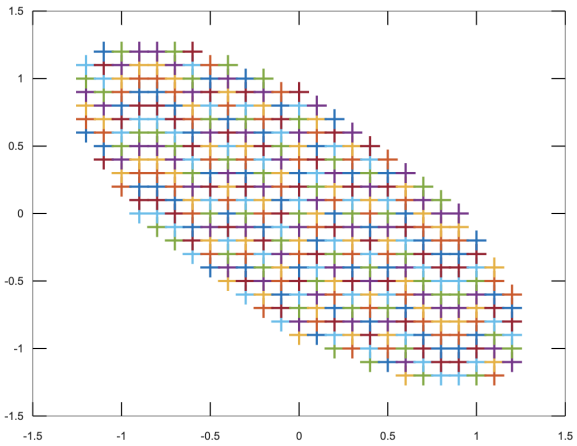


FIGURE 7 – Visualisation de la zone de stabilité

Code

```
A=[u0*(1-mu), 1; 1, -u0*4*(1-mu)];

B=[mu+(1-mu)*x10; mu-4*(1-mu)*x20];

%TODO avec riccati, trouver une commande stabilisante
%essai riccati : A'P+PA-PB inv(R) B'P + Q =0
[x, l, g] =  care(A,B,Q,R);
K=-g

%TODO calculer l'??quation du systeme avec rebouclage
%systeme rebouclage
Ak=A+B*K;
%eigs(Ak);

M=[-1,0;0,-1] - (Ak);

%TODO calculer la borne lambda et la borne alpha a 95 % de lambda
%calcul de la borne, on retrouve bien celle de l'article
lambda= -max(eigs(Ak));
% borne a 95 %
alpha= lambda - 0.05*lambda;

%TODO ecrire les matrices de l'??quation de Lyapunov et la resoudre pour
    obtenir la matrice P
%matrice pour equation lyap
Al= (Ak+[alpha,0;0,alpha])';
Bl=(Q+K'*R*K);

P=lyap(Al,Bl)

%ici on calcul la borne du probleme quadratique beta
[x1,obj]=qp([0;0],-2*P,[],[],[],[-0.8;-0.8],[0.8;0.8],-2,K,2)
%MATLAB
%obj = -1.0486;
beta=-obj;

%TODO ecrire le test qui valide ou non si le point est dans la zone de
    stabilit?? du controleur
test=x_verif'*P*x_verif;
ok = (test < beta);
```

5 Commande prédictive

Linéarisation : La commande stabilisante A_r et B_r est calculée par l'équation 2 et 3.

$$\begin{aligned} A_r &= \begin{bmatrix} u_0(1-\mu) & 1 \\ 1 & -4u_0(1-\mu) \end{bmatrix} \\ B_r &= [\mu + (1-\mu)x_1(0), \mu - 4(1-\mu)x_2(0)] \end{aligned} \tag{4}$$

Discrétisation :

$$\begin{aligned} A &= I + A_r \delta t \\ B &= B_r \delta t \end{aligned} \tag{5}$$

Vectorisation :

$$\begin{aligned} A_{qp} &= [A, A^2, A^3, A^4] \\ B_{qp} &= \begin{bmatrix} B & 0 & 0 & 0 \\ AB & B & 0 & 0 \\ AAB & AB & B & 0 \\ AAAB & AAB & AB & B \end{bmatrix} \end{aligned} \tag{6}$$

On a le système :

$$X = A_{qp} x + BqpU \tag{7}$$

Le but est de trouver la commande prédictive U optimale tel que $X = 0$ (Erreur à la trajectoire nulle).

$$U^* = -B^{\#} A_{qp} x \tag{8}$$

D'abord, pour vérifier que la commande est vraiment stabilisante, nous exécutons le code *simulateMPC.m* avec la matrice du régulateur obtenue précédemment : $K = \begin{bmatrix} 2.118 & 2.118 \end{bmatrix}$ et à un point initial dans l'ellipse de stabilité [-0.523 ;0.244] ($\mu = 0.5$). La solution est stable et converge bien vers l'état [0 ;0].

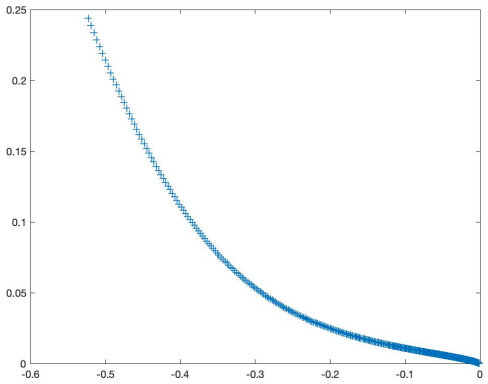
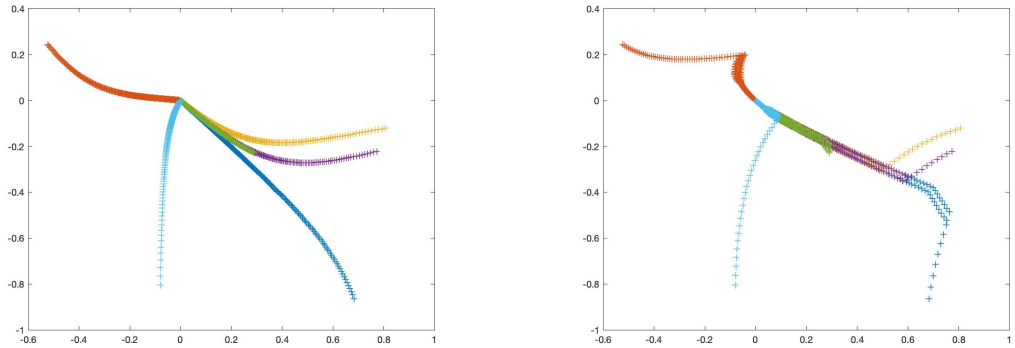


FIGURE 8 – Solution stable



(a) Commande stabilisante (b) Commande prédictive

FIGURE 9 – Commande stabilisante et commande predictive

Dans (a) nous trouvons les trajectoires relatives à la simulation en commande stabilisante pour différentes positions initiales dans l'ellipse de stabilité. Nous remarquons que effectivement les solutions convergent vers l'état 0.

Pour la commande prédictive en (b), nous prenons un horizon de 4, Et nous voyons que les solution obtenues son de tel sorte à minimiser la fonction de coût qui est proportionnelle à la somme de la norme de l'état et de la commande sur l'horizon de prédiction. Donc le système suit une trajectoire qui minimise la norme de la commande et l'état vis à vis à la dynamique du système, et converge vers l'état 0 (stable), mais peut osciller au début.

Code

```
%TODO linearisation en x et t
A=eye(2,2)+dt*[u*(1-mu),1;1, -u*4*(1-mu)];
B=dt*[mu+(1-mu)*x(1);mu-4*(1-mu)*x(2)];

%vecteur d'entrees
U=[u,u,u,u]';
H=eye(n,n)*2;


%TODO ecrire les matrices de la commande pr??dictive lin??aire
Aqp=[A;A*A;A*A*A;A*A*A*A];

Bqp=[B, zeros(2,n-1);
    A*B,B,zeros(2,n-2);
    A*A*B, A*B, B, zeros(2, n-3);
    A*A*A*B, A*A*B, A*B, B];

%TODO avec une pseudo inverse, calculer le vecteur d'entr??es
U=-inv(Bqp'*Bqp)*Bqp'*Aqp*x;
```

Références

[1] Hong Chen et Frank Allgöwer. "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability". In : *Automatica* 34.10 (1998), p. 1205-1217.