ROB316 - Planification et contrôle
TP2 - Commande avec anticipation, commande prédictive

COELHO RUBACK Arthur
SANTOS SANO Matheus

Palaiseau, France

Décembre 2023

Résumé Cours

Le cours présente le modèle dynamique qui, contrairement au modèle cinématique, prend en compte les forces et moments présents dans le système. Le problème dynamique direct est celui dans lequel le calcul des accélérations provient des efforts donnés, tandis que dans le problème dynamique indirect, les efforts sont calculés à partir du mouvement.

Dans ce cours, on présente la technique de contrôle appelée commande prédictive (MPC). Le MPC prédit l'état du système dynamique et calcule une séquence de commandes en boucle ouverte pour un coût minimal et en respectant des contraintes. De cette façon, le MPC fait le système suivre une consigne, en minimisant les coûts et en respectant les contraintes.

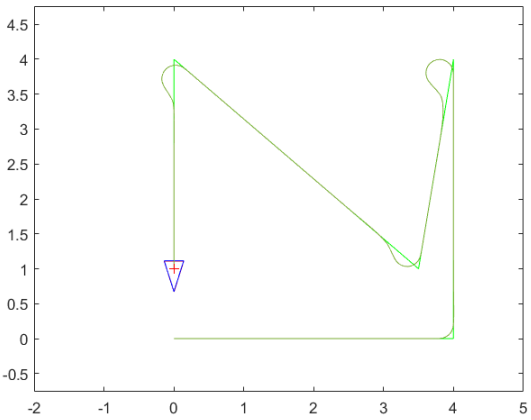
Enfin, il est présenté comment implémenter la commande prédictive. En bref, l'approche se découpe en étapes : linéarisation du modèle dynamique non-linéaire, discrétisation du système par le schéma d'Euler, son vectorisation et résolution de ce système vectorisé.

Introduction TP

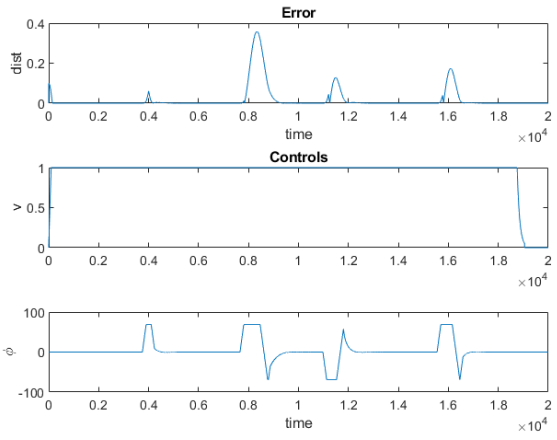
Ce TP est divisé en trois parties. Dans la première partie, la méthode MPC est implémentée et le contrôle du robot est comparé avec MPC et PID. De plus, l'influence de la taille de l'horizon sur le contrôle du vélo est étudiée. Dans la deuxième partie, on implémente la méthode pour obtenir la zone de stabilité d'une commande prédictive. Dans la partie finale, la commande prédictive est implémentée et 6 simulations sont exécutées pour comparer les résultats obtenus par la commande prédictive et la commande de stabilisation (obtenue dans la deuxième partie).

Question 1 - anticipation

Dans cette première étape, on compare les résultats du contrôle d'un robot par PID et par commande prédictive (MPC). Pour cela, on compare le résultat obtenu par PID (Figure 1) et le résultat obtenu par MPC (Figure 2). Il est à noter que on fixe les gains $K_\rho = 10$ et $K_\alpha = 5$.

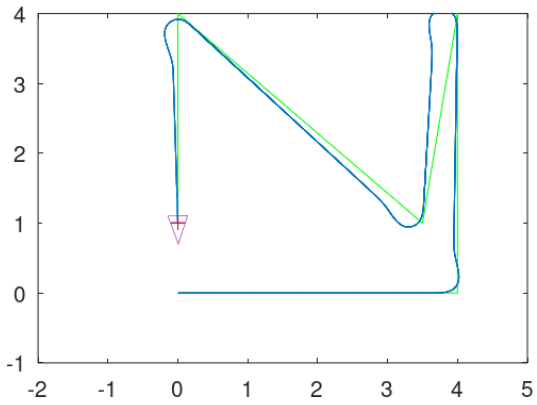


(a) Trajectoire du robot contrôlé par PID.

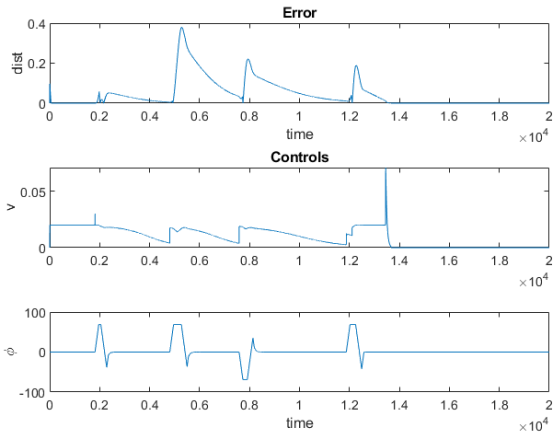


(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par PID.

FIGURE 1 – Robot contrôlé par PID avec $K_\rho = 10$ et $K_\alpha = 5$.



(a) Trajectoire du robot contrôlé par MPC.



(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par MPC.

FIGURE 2 – Robot contrôlé par MPC avec $K_\rho = 10$, $K_\alpha = 5$ et $window\_size = 5$.

Comme on peut le voir dans les figures ci-dessus, la trajectoire réalisée par le robot contrôlé par PID est plus proche que la trajectoire du robot contrôlé par la méthode MPC. Par conséquent, l'erreur par MPC est logiquement plus grande que celle obtenue dans le cas du PID, comme le montre le Tableau 1.

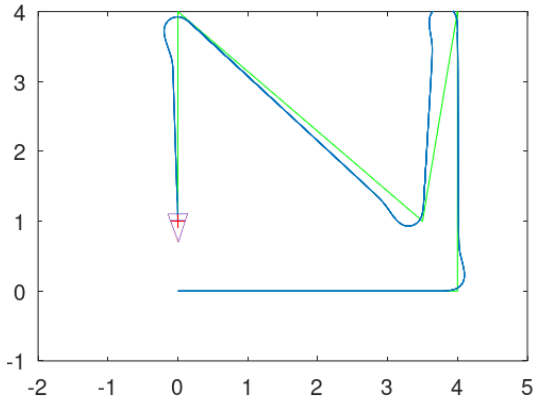
Erreur	
PID	MPC
$367,9014$	$866,5697$

TABLE 1 – Erreur par PID et par MPC.

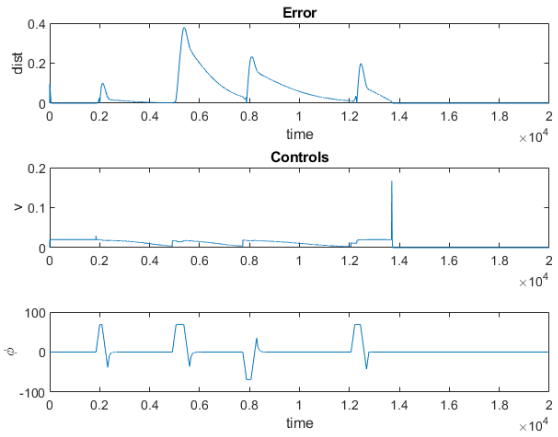
La méthode PID contrôle le robot par rapport à la trajectoire de référence. Le contrôle PID s'efforce donc de faire suivre au robot les points de la trajectoire de référence. En outre, le contrôleur PID est capable d'ajuster rapidement les sorties proportionnelles du contrôleur et, ainsi, de réagir plus rapidement aux courbes et de faire en sorte que le robot suive principalement la trajectoire de référence.

D'autre part, le MPC est une méthode de contrôle à long terme qui prédit le comportement futur du système et donc contrôle le robot de manière à ce qu'il puisse atteindre l'objectif final de manière optimale, sans nécessairement suivre la trajectoire de référence. Par conséquent, comme le montre la Figure 2, après une courbe, le robot s'éloigne de la trajectoire de référence et, au fil du temps, il s'en rapproche à nouveau.

Cette caractéristique du MPC devient plus évidente lorsque l'on augmente le nombre de points sur l'horizon ($window\_size$). En augmentant le nombre de points sur l'horizon, le contrôle par MPC considérera un plus grand nombre de points futurs le long de la trajectoire. Ainsi, le robot sera contrôlé sur la base d'un grand nombre de points. Comme la méthode MPC cherche à contrôler le robot de manière à ce qu'il arrive à l'objectif final de manière plus optimale, en augmentant $window\_size$, le robot suivra moins la trajectoire de référence et se dirige vers la position finale.

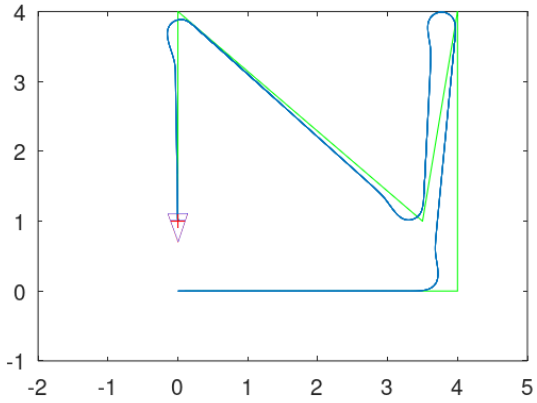


(a) Trajectoire du robot contrôlé par MPC.

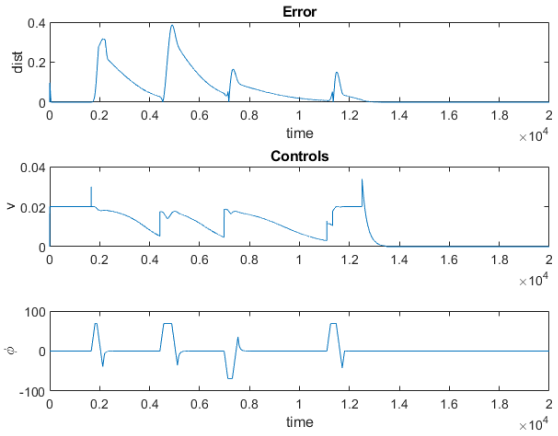


(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par MPC.

FIGURE 3 – Robot contrôlé par MPC avec $window\_size = 1$.



(a) Trajectoire du robot contrôlé par MPC.



(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par MPC.

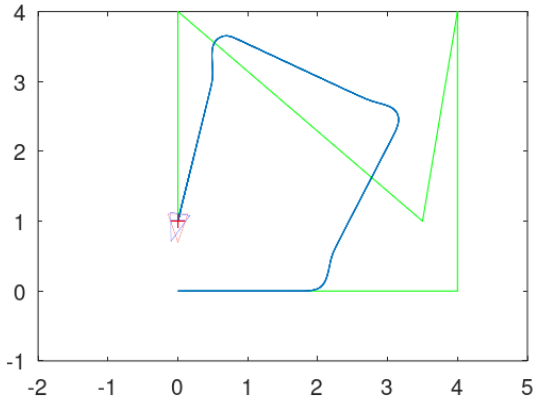
FIGURE 4 – Robot contrôlé par MPC avec $window\_size = 20$.

Comme le montrent les Figures 5 et 6, plus la *window_size* est grande, plus le robot a une vision complète de la trajectoire. Dans ce cas, le robot est contrôlé par la méthode MPC pour aller directement au objectif final, sans suivre la trajectoire de référence. Ainsi, quand *window_size* est grande, l'erreur est aussi importante, comme le montre le Tableau 2.

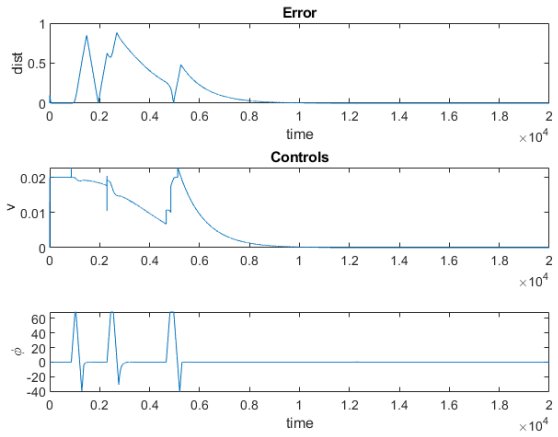
Erreur par rapport à *window_size*				
1	5	20	100	1000
867, 1894	866, 5697	1014, 8871	2405, 0702	1687, 9893

TABLE 2 – Erreur du MPC avec *window_size* égal à 1, 5, 20, 100 et 1000.

D'autre part, si la *window_size* est très petite, le nombre de points qui seront pris en compte lors du contrôle du robot est également très petit, de sorte que le contrôle anticipe moins la trajectoire, comme on peut le voir sur la Figure 3.

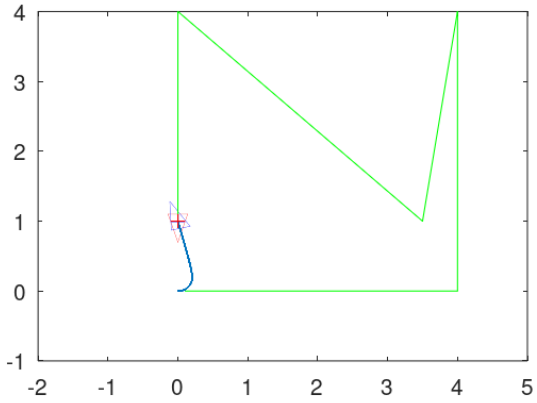


(a) Trajectoire du robot contrôlé par MPC.

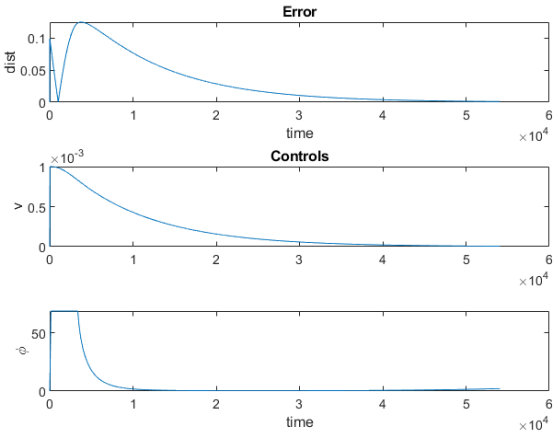


(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par MPC.

FIGURE 5 – Robot contrôlé par MPC avec $window\_size = 100$.



(a) Trajectoire du robot contrôlé par MPC.



(b) Erreur de position et direction, et variables de contrôle de vitesse du robot par MPC.

FIGURE 6 – Robot contrôlé par MPC avec $window\_size = 1000$.

Question 2 - zone de stabilité d'une commande prédictive

Dans cette deuxième partie, on implémente une fonction qui vérifie si un point se trouve dans la zone de stabilité d'un commande prédictive en utilisant la méthode présentée dans l'article "A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability" de H. Chen et F. Allgöwer. Le système est décrit par le système d'EDO suivant :

$$\begin{cases} \dot{x}_1 = x_2 + u(\mu + (1-\mu)x_1) \\ \dot{x}_2 = x_1 + u(\mu - 4(1-\mu)x_2) \end{cases} \tag{1}$$

Ce système est instable, il faut donc le linéariser pour qu'il se stabilise pour tout $\mu \in (0, 1)$. Il faut noter qu'après la linéarisation, le système n'est pas contrôlable. En linéarisant le système (1), on obtient :

$$\tilde{x}_1 = x_1 + x_{10}$$
$$\tilde{x}_2 = x_2 + x_{20}$$
$$\begin{cases} \dot{\tilde{x}}_1 = \tilde{x}_2 + u(1-\mu)\tilde{x}_1 + u(\mu + (1-\mu)x_{10}) + x_{20} \\ \dot{\tilde{x}}_1 = \tilde{x}_2 - 4u(1-\mu)\tilde{x}_2 + u(\mu - 4(1-\mu)x_{20}) + x_{10} \end{cases}$$

Ainsi, la linéarisation jacobienne du système à l'origine est considérée comme :

$$\dot{x} = Ax + Bu \tag{2}$$

où les matrices A et B sont définies comme suit :

$$A = \begin{pmatrix} u(1-\mu) & 1 \\ 1 & -4u(1-\mu) \end{pmatrix} \quad B = \begin{pmatrix} \mu + (1-\mu)x_{10} \\ \mu - 4(1-\mu)x_{20} \end{pmatrix}$$

En utilisant la fonction *care* d'Octave, on résout l'équation de Riccati, avec les matrices de pondération dans l'objectif fonctionnel :

$$Q = \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix} \quad R = \begin{pmatrix} 1 \end{pmatrix}$$

En résolvant l'équation de Ricatti à l'aide de la fonction *care*, on obtient la matrice K. Comme l'équation (2) est stabilisable, un feedback d'état linéaire $u = Kx$ peut être déterminé de telle sorte que $A_k := A + BK$ soit asymptotiquement stable. Ainsi, la borne λ est la valeur propre maximale de A_k et que la borne alpha est 95% de λ.

Enfin, les matrices de l'équation de Lyapunov sont résolues à l'aide de la fonction *lyap* d'Octave et, ainsi, le point de la région de satisfaction des contraintes est trouvé à l'aide de la fonction *qp* d'Octave. La zone des points stables trouvés pour le système du robot est illustrée à la Figure 7. On peut voir la zone de stabilité avec toutes les points pour lesquels la commande prédictive est stabilisante.

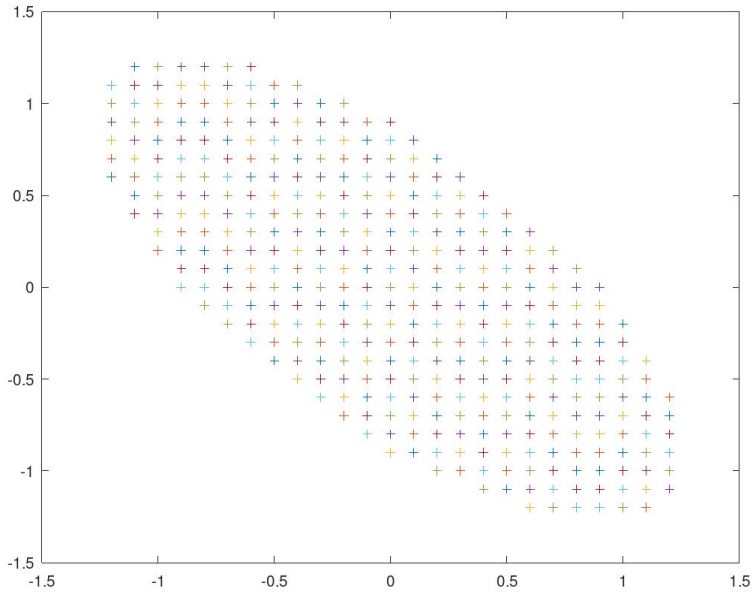


FIGURE 7 – Zone de stabilité.

Question 3 - commande prédictive

À partir de la commande de stabilisation calculée dans la section précédente ($K = [2,118 \quad 2,118]$), on peut voir sur la Figure 8 que tous les points des différents emplacements convergent vers le point de référence 0. Il convient de souligner que cette convergence était possible parce que tous les points se trouvaient à l'intérieur de la zone de stabilité.

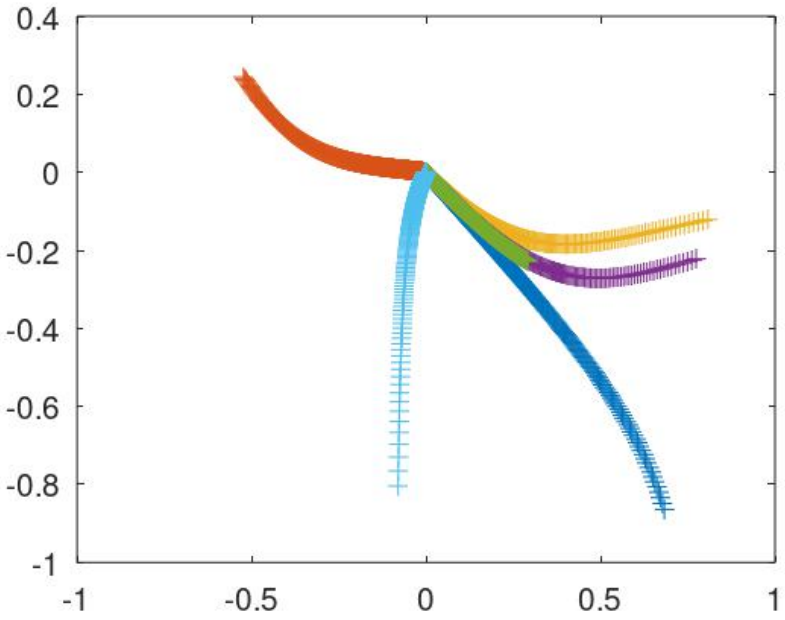


FIGURE 8 – Résultats de la simulation avec la commande de stabilisation.

Le calcul de la commande prédictive simplifiée pour un système non linéaire est divisé en plusieurs étapes. Tout d'abord, le système est linéarisé, puis discrétisé à l'aide du schéma d'Euler. Ensuite, le système est vectorisé, ce qui permet d'obtenir un seul système pour l'optimisation à l'aide de l'approche des moindres carrés. Le système vectorisé à résoudre est le suivant :

$$X(k) = \hat{A}\tilde{x}(k) + \hat{B}U$$
$$U = -\hat{B}^{\#}\hat{A}\tilde{x}(k)$$

Comme on considère un horizon de 4, les matrices A et B deviennent :

$$A = \begin{pmatrix} A(k) \\ A^2(k) \\ A^3(k) \\ A^4(k) \end{pmatrix} \qquad B = \begin{pmatrix} B(k) & 0 & 0 & 0 \\ A(k)B(k) & B(k) & 0 & 0 \\ A^2(k)B(k) & A(k)B(k) & B(k) & 0 \\ A^3(k)B(k) & A^2(k)B(k) & A(k)B(k) & B(k) \end{pmatrix}$$

À partir de cette commande prédictive simplifiée, on obtient les résultats de la simulation présentés dans la Figure 9.

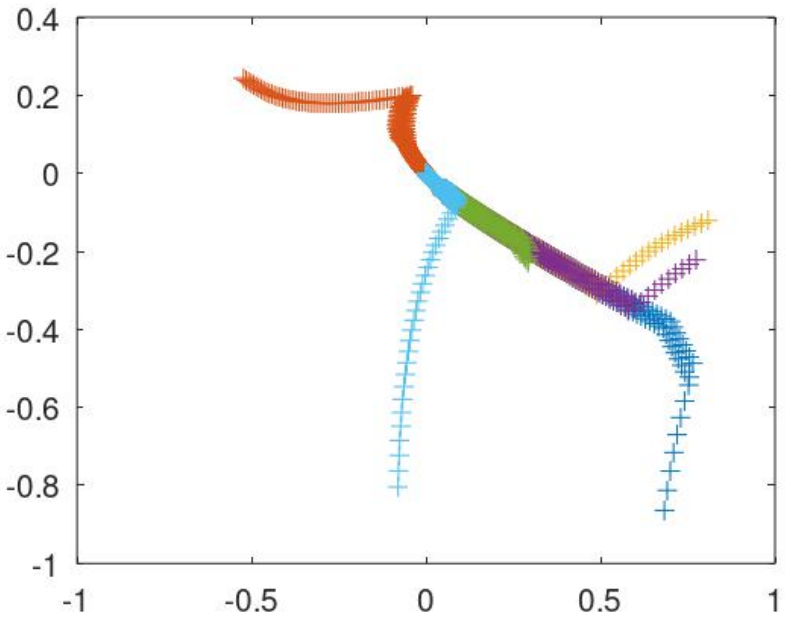


FIGURE 9 – Résultats de la simulation avec la commande prédictive simplifiée.

En comparant les deux cas (Figure 8 et Figure 9), ou peut voir que les résultats obtenus pour la commande de stabilisation (Figure 8) sont meilleurs que ceux obtenus par la commande prédictive simplifiée (Figure 9). Cependant, les deux commandes fonctionnent correctement et sont capables de converger vers l'objectif final (position $[0, 0]$).