TP4: Commande avec anticipation, commande prédictive
RO16 - Planification et control

Gallego, Natalia
AST
ENSTA Paris
natalia.gallego@ensta.fr

I. Introduction

Dans ce TP, nous explorons les concepts de contrôle anticipatif et de contrôle prédictif à travers des exercices basés sur un modèle de robot de type vélo.

Ce TP est structuré en trois parties principales : Dans un premier temps, nous analyserons l'effet de l'anticipation sur une trajectoire en implémentant une commande qui calcule une séquence de points dans le futur, au lieu d'un seul point comme dans le travail précédent. Nous examinerons l'impact de différents horizons d'anticipation.

D'autre part, nous déterminerons si un point donné se situe dans la zone de stabilité d'un contrôleur. Cette partie se concentre sur un système dynamique donné et un contrôleur borné. Nous utiliserons une commande calculée pour vérifier la stabilisation du système.

Enfin, nous mettrons en œuvre un contrôle prédictif simplifié avec un horizon en 4 étapes. Des simulations seront réalisées avec un contrôle stabilisant ou prédictif et nous comparerons leurs performances pour différents points initiaux dans la zone de stabilité.

II. Question 1: Anticipation

Dans ce premier exercice nous allons étudier l'effet de l'anticipation sur une trajectoire. L'idée est d'anticiper le chemin en calculant une série de points (au lieu d'un seul point).

Pour réaliser cette procédure, le code **BicycleToPathControl2** a été modifié.

Pour le système d'anticipation, deux variables persistantes sont utilisées, l'index du point d'objectif actuel sur le chemin (`goalWaypointId`) et les coordonnées du point d'objectif actuel (`xGoal`). itemize

D'autre part, il existe également des variables telles que la portée d'influence (ρ), qui est le rayon dans lequel le robot considère qu'il a atteint un point cible et passe au suivant. De plus, la fenêtre d'anticipation (`window_size`) définit le nombre de points futurs considérés dans la trajectoire.

Pour construire la liste des points futurs, le code part de la position actuelle du robot. Si le robot est à portée ρ d'un point cible, il est considéré comme atteint, ajouté à `list_points` et passé à le prochain point sur le chemin.

S'il n'a pas atteint le point cible, il avance dans la direction normalisée vers le point cible sur une distance $d_{\max}$:

Ce processus est répété jusqu'à ce que les points `window_size` dans `list_points` soient remplis.

Le contrôle est effectué comme le TP précédent, où deux erreurs sont calculées (erreur de distance et erreur angulaire). Cela génère un contrôle selon :

Le contrôle généré est :

$$v = K_\rho \cdot \frac{e_{\text{goal\_dist}}}{\texttt{window\_size} \cdot 10} \tag{1}$$

$$\phi = K_\alpha \cdot e_{\text{angle}}, \tag{2}$$

où K_ρ et K_α sont les gains proportionnels pour la vitesse linéaire et angulaire, avec des valeurs de **10** et **5** respectivement.

A. Horizon à 5 points et comparaison avec le contrôle TP précédent

Ce qui suit est une comparaison des résultats d'un contrôle de trajectoire sans anticipation, tel que celui réalisé dans le TP précédent, et d'un contrôle avec anticipation, tel que celui auquel nous avons affaire actuellement. Dans le cas du TP précédent, la fonction est utilisée pour créer une sous-trajectoire avec des points interpolés uniformément répartis le long de la trajectoire d'origine ; Cela amène le robot à se déplacer vers un seul point cible sélectionné à partir de ce sous-chemin. D'autre part, une liste de points (`list_points`) est actuellement construite dynamiquement, avançant pas à pas de la position courante vers les points cibles. Cela implique que les points ne sont pas répartis uniformément, mais dépendent du mouvement du robot.

Les résultats de cette comparaison peuvent être vus dans les figures 1 et 2. Il convient de noter que la comparaison a été faite avec un horizon de valeur de 5.

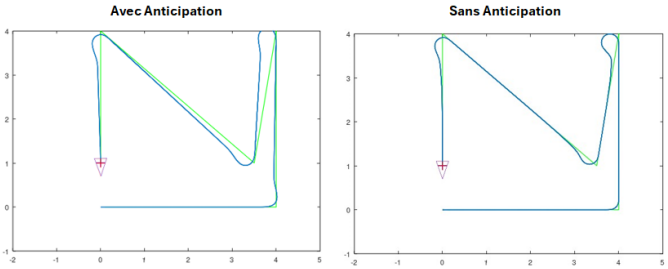

Fig. 1. Comparaison de trajectoire avec et sans anticipation

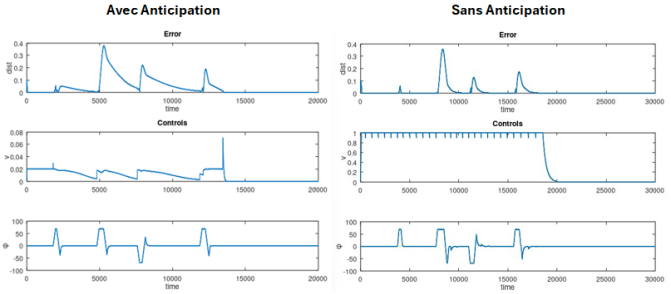

Fig. 2. Comparaison de l'erreur avec et sans anticipation

Grâce à l'anticipation, la trajectoire tend à se dérouler de manière plus fluide, puisqu'elle prend en compte plusieurs points futurs au lieu de réagir uniquement au point immédiat. Alors que le cas sans anticipation montre des mouvements plus brusques ou réactifs, notamment dans les parties où les courbes sont prononcées.

Cependant, en comparant les erreurs de suivi de trajectoire, alors que dans le scénario avec anticipation nous avons atteint une valeur de **867,7** dans le cas sans anticipation, contrairement à ce qui était attendu, une valeur inférieure égale à **436,4** a été obtenue.

B. Horizons: 1, 5, 20, 100, 1000

Afin de continuer à expérimenter le contrôle à l'avance, il a été décidé de revoir comment il varie en fonction de la valeur de l'horizon défini. Pour cela, les valeurs **1, 5, 20, 100** et **1000** ont été utilisées. Les résultats obtenus peut être vu dans les figures 3 et 4.

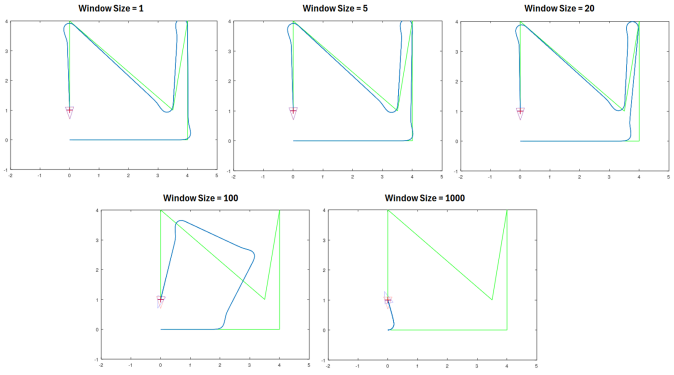

Fig. 3. Comparaison de trajectoire en fonction de la valeur de l'horizon

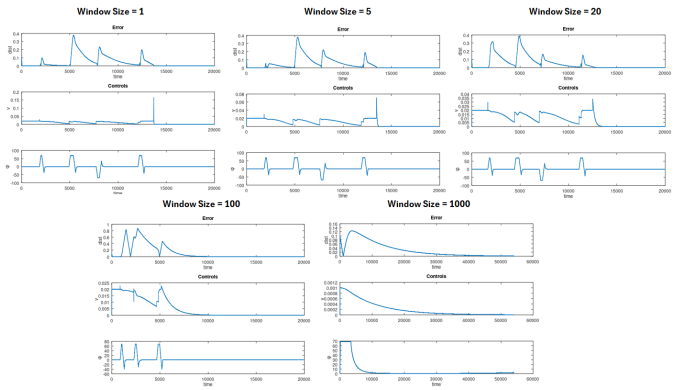

Fig. 4. Comparaison de l'erreur en fonction de la valeur de l'horizon

L'horizon de prévision (`window_size`) détermine le nombre de points futurs du chemin à prendre en compte lors du calcul du contrôle du robot.

Taille d'horizon 1 : Le robot ne considère que le point cible immédiat et n'anticipe pas les points futurs. Cela entraîne des mouvements plus réactifs et brusques, car le robot ajuste son contrôle uniquement en fonction de l'erreur vers le point immédiat suivant. Un suivi plus précis est toutefois obtenu, mais des problèmes surviennent dans les courbes très serrées.

Taille Horizon 5 : Améliore la fluidité du mouvement lorsque le robot commence à anticiper légèrement la direction du chemin. Oscillations réduites, notamment sur les trajectoires droites ou légèrement courbes.

Horizon taille 20 : L'anticipation devient significative. Le robot planifie ses mouvements en fonction de la tendance générale du chemin plutôt que de répondre à chaque point immédiat. Les mouvements sont beaucoup plus fluides, avec moins de probabilité d'oscillations même sur des trajectoires courbes. Cependant, cela conduit à commencer à couper les coins ronds en raison d'une planification basée sur un horizon plus large.

Horizon taille 100 : Il s'agit d'une planification extrêmement précoce. Le robot optimise ses mouvements en fonction d'un long tronçon du parcours. De ce fait, on

a tendance à ignorer les détails précis de la trajectoire, en privilégiant les mouvements globaux. Sur les trajectoires courbes, le robot dévie considérablement ou prend des raccourcis.

Horizon taille 1000 : Le robot trace une ligne droite vers le but final plutôt que de suivre chaque détail du chemin. Cette approche ignore la plupart des points intermédiaires. Cela amène le robot à générer des mouvements optimisés globalement mais avec peu d'adhérence à la trajectoire.

De plus, les valeurs d'erreur de chacune des configurations ont été prises, elles sont présentées dans le tableau II-B

L'Horizons	L'Erreur
1	867.07
5	865.93
20	1014.93
100	2405.07
1000	1687.99

III. Question 2: Zone de stabilité d'une commande prédictive

Dans cet exercice, la fonction $verify\_stability$ a été écrite. Cette fonction cherche à vérifier si un point appartient à la région de stabilité d'un contrôleur basé sur Lyapunov.

Les équations dynamiques du système sont :

$$\dot{x_1} = x_2 + u(\mu + (1-\mu)x_1) \tag{3}$$

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

Pour cela pour un système non linéaireson modèle linéarisé sera obtenu autour d'un point d'équilibre (x_{10}, x_{20}) en utilisant les Jacobiennes. Donc, les matrices A et B doivent être trouvées:

$$A = \begin{bmatrix} \frac{\partial \dot{x_1}}{\partial x_1} & \frac{\partial \dot{x_1}}{\partial x_2} \\ \frac{\partial \dot{x_2}}{\partial x_1} & \frac{\partial \dot{x_2}}{\partial x_2} \end{bmatrix} = \begin{bmatrix} (1-\mu)u & 1 \\ 1 & -4(1-\mu)u \end{bmatrix} \tag{5}$$

$$B = \begin{bmatrix} \frac{\partial \dot{x_1}}{\partial u} \\ \frac{\partial \dot{x_2}}{\partial u} \end{bmatrix} = \begin{bmatrix} \mu + (1-\mu)x_{10} \\ \mu - 4(1-\mu)x_{20} \end{bmatrix} \tag{6}$$

Ces matrices définissent le comportement du système autour du point d'équilibre.

L'objectif ultime est de concevoir un contrôleur d'état qui minimise un coût quadratique. Pour cela, l'équation algébrique de Riccati est utilisée et avec cela nous obtenons le gain résultant est:

$$K = -\begin{bmatrix} 2.118 & 2.118 \end{bmatrix} \tag{7}$$

Le système en boucle fermée,(noté A_k), est défini comme:

$$A_k = A + BK \tag{8}$$

Enfin, pour vérifier la stabilité, les equations de Lyapunov doivent être résolues. Pour assurer un taux de convergence, un paramètre $\alpha = 0,95\lambda_{max}$ est introduit où λ_{max} est la plus grande des entrées.

Pour résoudre cela avec Octave l'expression est prévue :

$$AX + XA^\top + B = 0, \tag{9}$$

où $\mathbf{X}$ est l'inconnue à calculer. Dans ce cas, $\mathbf{P}$ représente la matrice inconnue à déterminer, tandis que $\mathbf{A}$ et $\mathbf{B}$ correspondent aux matrices de Lyapunov spécifiques.

L'équation de Lyapunov associée au système peut être exprimée comme suit :

$$(A_k + \alpha I)^\top P + P(A_k + \alpha I) + (Q + K^\top RK) = 0. \tag{10}$$

A partir de cette formulation, les matrices impliquées prennent la forme :

$$A_l = A_k + \alpha I, \quad B_l = Q + K^\top RK. \tag{11}$$

En résolvant cette équation, on obtient la matrice $\mathbf{P}$, qui permet d'évaluer la région de stabilité du système en termes de contrôle proposé.

Finalement, la zone de stabilité est définie comme les points $\mathbf{x}$ qui satisfont :

$$x^\top Px < \beta, \tag{12}$$

où β est calculé en résolvant un problème quadratique soumis à des contraintes :

$$\min x^\top Px, \quad \text{sous réserve de : } -0,8 \leq x_1, x_2 \leq 0,8. \tag{13}$$

On vérifie si le point appartient à la zone de stabilité. Si test $< \beta$, le point appartient à la zone de stabilité et le système est stable.

Les résultats obtenus sont présentés dans la figure 5.

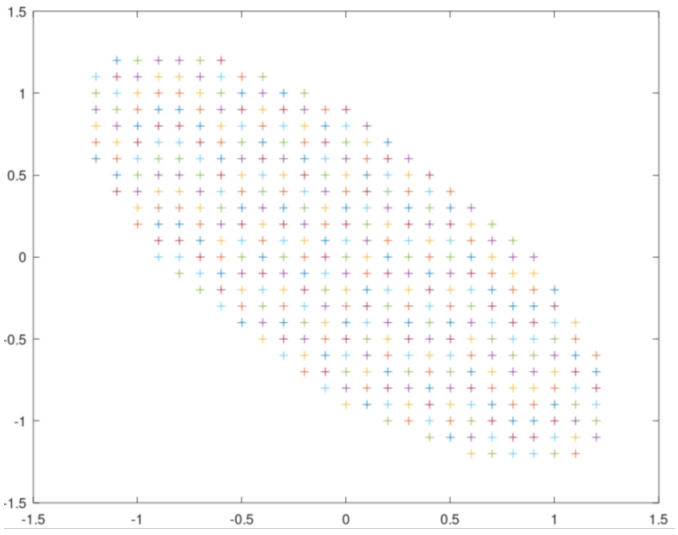

Fig. 5. Zone de stabilité de la commande predictive

IV. Question 3: Commande prédictive

Dans ce scénario, nous cherchons à tester une commande prédictive comme celle vue en classe. Pour cela, on utilise la fonction $test\_mpc$ qui lance 6 simulations à partir de points pris dans la zone de stabilité. Ces points sont :

$$Points = \begin{bmatrix} 0.683 & -0.864 \\ -0.523 & 0.244 \\ 0.808 & -0.121 \\ 0.774 & -0.222 \\ 0.292 & -0.228 \\ -0.08 & -0.804 \end{bmatrix} \tag{14}$$

Par contre, la matrice K va prendre deux valeurs pour faire deux expériences différentes. Tout d'abord, on utilisera celui calculé précédemment, ce qui permet à la sortie de commande d'être stable ; D'autre part, un tableau vide qui sera rempli avec une commande prédictive qui sera construite dans la fonction **simulerMPC**.

A. Commande stabilisante calculée en Q2

En utilisant les valeurs de la matrice de contrôle K obtenues dans l'exercice précédent :

$$K = -\begin{bmatrix} 2.118 & 2.118 \end{bmatrix} \tag{15}$$

Les résultats sont obtenus à partir de la figure 6. Comme on peut le voir sur cette figure, comme prévu, chacune des positions sources étudiées converge autour du point $[0, 0]$ avec un pas de plus en plus petit lié à la résolution quadratique du problème.

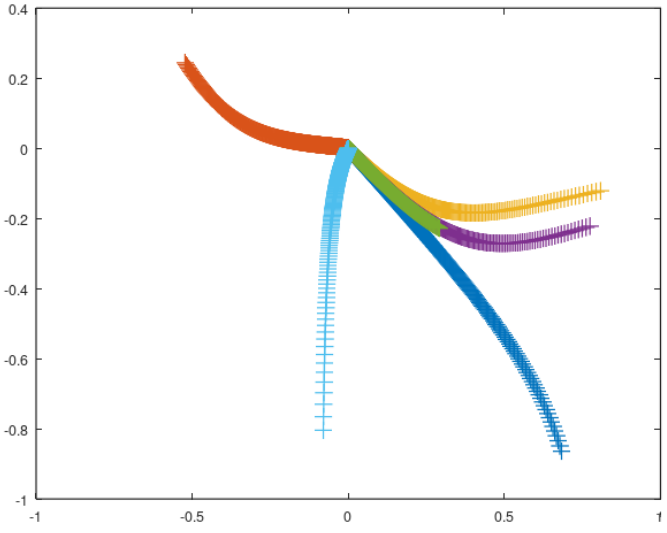

Fig. 6. Trajectoires avec les résultats du Q2

B. Commande Prédictive avec horizon de valeur 4

La fonction `simulateMPC` a été modifiée de manière à implémenter une simulation utilisant le contrôle prédictif du modèle (MPC) via une approche de prédiction de trajectoire sur un horizon temporel défini. Le MPC cherche à minimiser une fonction de coût sur un horizon de prédiction pour contrôler efficacement le système.

La première étape du calcul de contrôle prédictif est la linéarisation du système non linéaire. Les équations du système sous forme non linéaire sont :

$$\dot{x}_1 = x_2 + u \cdot (\mu + (1-\mu)x_1) \tag{16}$$

$$\dot{x}_2 = x_1 + u \cdot (\mu - 4(1-\mu)x_2) \tag{17}$$

L'objectif est d'obtenir une approximation linéaire du système autour d'un point de fonctionnement. À cette fin, une approximation d'Euler est utilisée, ce qui permet d'exprimer le système comme :

$$\hat{x}(k+1) = \hat{x}(k) + A(k) \cdot \hat{x}(k) + B(k) \cdot u(k) \tag{18}$$

Où les matrices $A(k)$ et $B(k)$ sont respectivement les matrices d'état et de transition d'entrée et sont définies comme suit :

$$A(k) = I + \Delta t \cdot \begin{bmatrix} (1-\mu)u & 1 \\ 1 & -4(1-\mu)u \end{bmatrix} \tag{19}$$

$$B(k) = \begin{bmatrix} \mu + (1-\mu)x_{10} \\ \mu - 4(1-\mu)x_{20} \end{bmatrix} \tag{20}$$

Une fois le système linéarisé, la technique de vectorisation est utilisée pour exprimer l'évolution du système sur plusieurs étapes de prédiction. Cela implique le calcul des matrices $\hat{A}$ et $\hat{B}$, qui contiennent les itérations des matrices $A(k)$ et $B(k)$ sur l'horizon de prédiction de 4 jours. étapes. Ces matrices sont calculées comme :

$$\hat{A} = \begin{bmatrix} A(k) \\ A(k)^2 \\ A(k)^3 \\ A(k)^4 \end{bmatrix} \tag{21}$$

$$\hat{B} = \begin{bmatrix} B(k) & 0 & 0 & 0 \\ A(k)B(k) & B(k) & 0 & 0 \\ A^2(k)B(k) & A(k)B(k) & B(k) & 0 \\ A^3(k)B(k) & A^2(k)B(k) & A(k)B(k) & B(k) \end{bmatrix} \tag{22}$$

Ces matrices représentent l'évolution des états et des entrées du système au cours de l'horizon de prédiction.

Une fois les matrices $\hat{A}$ et $\hat{B}$ obtenues, le contrôle prédictif U est calculé en utilisant une solution des moindres carrés. La formule est :

$$U = -(Bqp^T Bqp)^{-1} Bqp^T Aqp \tag{23}$$

Ce calcul vise à minimiser l'écart des états sur l'horizon de prédiction, garantissant ainsi que le système se stabilise au point d'équilibre souhaité.

La figure 7 montre les 6 simulations de trajectoire réalisées avec le contrôle prédictif. On peut observer que, comme

précédemment, les courbes convergent vers la position de référence, mais avec plus de difficulté, sans choisir le chemin le plus optimal en termes de longueur de trajet. De plus, nous pouvons voir que les courbes ont tendance à se déplacer le long d'un axe menant à la position de référence $[0, 0]$.

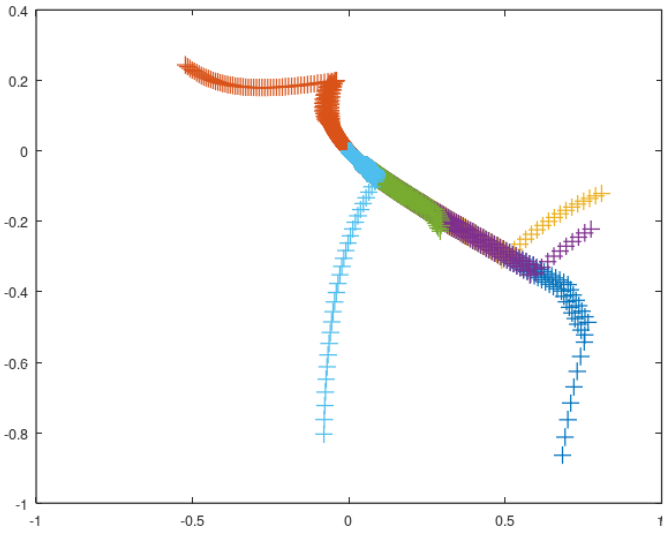

Fig. 7. Trajectoires avec commande prédictive

Nous voyons que les résultats obtenus en utilisant la solution de $Q2$ avec la matrice K sont meilleurs car cette approche a une plus grande capacité à stabiliser le système rapidement, avec une stratégie de contrôle direct qui garantit que le système converge vers la stabilité de la plupart manière efficace. En revanche, le contrôle prédictif présente une complexité de calcul plus élevée et est plus sensible aux erreurs de prédiction, ce qui peut conduire à une stabilisation plus lente, voire inefficace, si l'horizon de prédiction n'est pas adéquat (dans ce cas, valeur **4**).

V. Conclusions

L'utilisation de la fonction d'anticipation lors du calcul de plusieurs points le long du chemin améliore la précision et l'efficacité du suivi du chemin. À mesure que le nombre de points dans l'horizon d'anticipation augmente, le système commence à ignorer la trajectoire définie et recherche uniquement le chemin optimal pour atteindre le point final de l'itinéraire.

L'implémentation de la fonction de contrôle de stabilité montre que le système est instable pour certaines valeurs du paramètre μ, mais devient stabilisable lorsque la loi de commande est correctement calculée. Une stabilisation efficace dépend du choix approprié de la matrice de contrôle, que ce soit par rétroaction linéaire ou par contrôle prédictif.

La mise en œuvre d'une commande stabilisatrice utilisant la matrice K calculée en $Q2$ s'avère plus efficace en termes de vitesse de convergence face aux perturbations du système. Cela est dû au fait que le contrôle de stabilisation fournit une rétroaction constante et garantit que le système se stabilise rapidement. En comparaison, le contrôle prédictif nécessite un horizon de prédiction et peut être plus sensible aux erreurs de prédiction ou à la taille de l'horizon choisi, ce qui affecte ses performances et sa capacité de stabilisation.

VI. GitHub

Vous pouvez vous référer à la référence suivante pour voir les codes de nœud du projet: GitHub/RO16/TP4