Rapport de projet : Hovercraft control laws variant

BOUBKRAOUI Mustapha

02/2023

1 Introduction

Le projet "Hovercraft" fait partie de notre module "Commande pour la robotique". Il vise à explorer les concepts de liberté et de platitude pour renforcer les connaissances acquises en matière de commande de systèmes, en se concentrant sur les systèmes robotiques, tels que les aéroglisseurs. Les aéroglisseurs sont des engins hybrides qui peuvent se déplacer sur diverses surfaces, telles que la terre, l'eau et la glace, grâce à un coussin d'air haute pression généré entre la coque et la surface en dessous, contenu dans une "jupe" flexible. Le but du projet est de développer une trajectoire prescrite pour l'aéroglisseur en utilisant un modèle cinématique simplifié, afin de faciliter la modélisation des forces hydrodynamiques. Les aéroglisseurs sont utilisés dans de nombreuses applications industrielles, telles que les secours en cas de catastrophe, les opérations de garde-côtes, les enquêtes militaires et les sports nautiques.

2 Description du modèle cinématique

Pour construire les équations différentielles du modèle cinématique du Hovercraft, on présente l'éspace de mdélisation du système à décrire pour avoir une vue claire est globale sur le système. On remarque deux reférences, l'un est le référentiel globale (O-XYψ) et l'autre est un réferentiel fixe attachée au Hovercraft (o-uvr).

2.1 Modèle cinématique complet :

En utilisant le modèle décrit dans [?], on représente la position x,y et l'orientation ψ du système.

$$\begin{cases} \dot{x} = cos(\psi)u - sin(\psi)v \\ \dot{y} = cos(\psi)u + sin(\psi)v \\ \dot{\psi} = r \end{cases}$$

Sachant que (x,y,ψ) le référentiel de l'espace de Hovercraft, et (u,v,r) sont les vitesses de surge, sway et yaw.

On considére aprés de controler la position sans l'orientation ψ, en regard des équations précedents, pour simplifier les équations polynomial et eliminant ψ, on utilise la transformation des coordinations comme cité déja .

$$\begin{cases} z_1 = cos(\psi)x + sin(\psi)y \\ z_2 = -sin(\psi)x + cos(\psi)y \\ z_3 = \psi \end{cases}$$

On s'intéresse maintenant à faire une simplifaction des équations déja obtenue dans la partie précedente. On reçoit le module comme vu dans une littérature cité dans [?] :

$$\begin{cases} \dot{u} = vr + \tau_u \\ \dot{v} = -ur \\ \dot{r} = \tau_r \\ \dot{z}_1 = u + z_2 r \\ \dot{z}_2 = v - z_1 r \end{cases}$$

2.2 Modèle cinématique simplifié

Les dynamiques simplifié des précedentes équations de module sont les suivantes :

$$\begin{cases} \dot{u} = vr + \tau_u \\ \dot{v} = -ur \\ \dot{r} = \tau_r \end{cases}$$

Ou τ_u est la commande du surge et τ_r est la commande on yaw et (u,r) notre sortie plate.

3 Contrôle de suivi du système non lineaire :

3.1 Analyse de Platitude :

On prend le modèle cinématique simplifié de Hovercraft déja représenté par les équations s'état non linéaire déveveloppées dans la partie précédante.

Notre système est commandé par un vecteur de commande de é composantes : τ_u et τ_r, ce qui signifie que notre sortie plate si elle existe, elle admet 2 composantes.

3.1.1 Parametrisation :

Maintenant on cherche les équations qui representent les commandes τ_u et τ_r on fonctions des sorties plates uniquement :

$$\begin{cases} \tau_u = \dot{u} - vr = \dot{u} - v\frac{\dot{r}}{-u} \\ \tau r = \dot{r} = \frac{-\dot{u}\frac{\dot{v}}{u} - \ddot{v}}{u} \end{cases}$$

3.1.2 Algorithme d'extension dynamique :

On utilise l'algorithme d'extenstion dynamique pour montrer que y = (u,v) est notre sorties plates :

Phase I :

$$\begin{cases} \dot{u} = vr + \tau_u(\kappa_1 = 1) \\ \ddot{v} = -\dot{u}r - u\dot{r} = -\dot{u}r - u\tau_r(\kappa_2 = 2) \end{cases}$$

Phase II :

Puisque $\kappa_1 + \kappa_2 = 1 + 2 = 3 =$ dimension de l'état. le modéle admet alors (u,v) comme des sorties plates de notre système.

3.1.3 Closed loop trajectory tracking :

On choisisant (v1,v2) comme la nouvelle entrée, le bouclage par la suite est donnée par les équations suivantes :

on pose $y_r = (u_r, v_r)$ comme la trajectoire de la sortie plate y :

$$\begin{cases} v_1 = \dot{u}_r - \lambda_1(u - u_r) \\ v_2 = \ddot{v}_r - \lambda_2(v - v_r) - \lambda_3(\dot{v} - \dot{v}_r) \end{cases}$$

on représente les commandes avec les nouveles entrées choisis :

$$\begin{cases} \tau_u = v_1 - vr \\ \tau_r = \frac{-\dot{u}r - v_2}{u} \end{cases}$$

On remplaçant les dernières équations qu'on a trouvé pour (v1,v2), on trouve les commandes suivantes :

$$\begin{cases} \tau_u = -vr + \dot{u}_r - \lambda_1(u - u_r) \\ \tau_r = \frac{-\dot{u}r - \ddot{v}_r - \lambda_2(v - v_r) - \lambda_3(\dot{v} - \dot{v}_r)}{u} \end{cases}$$

3.2 Simulations et résultast

3.3 La performance du système sans perturbations

Dans cette partie, on va montrer les résultats des simulations de la commande synthétiser.
On definit les etats initiales comme suivant : $u_0 = 1$, $v_0 = 2$, $u_0 = 1$.
On fixant une trajectoire de réference sous forme d'une tangente hyperbolique de la forme suivante :

On obtient les résultats suivantes

On remarque que le trajectoire de l'etat u suive celle de l'etat de reference instantanément avec une marge d'erreur assez petite. Il y a une petite fluctiation au debut pour l'etat v, cela revient au

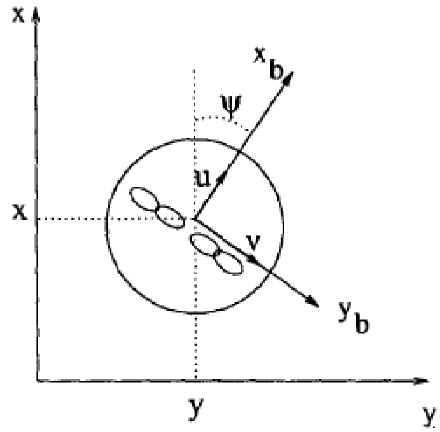

FIGURE 1 – The Hovercraft.

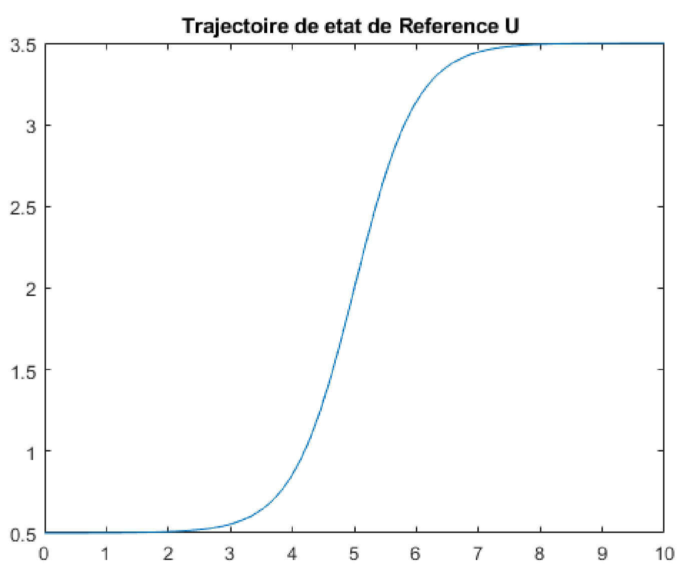

FIGURE 2 – Trajectoire de refeence pour l'état u

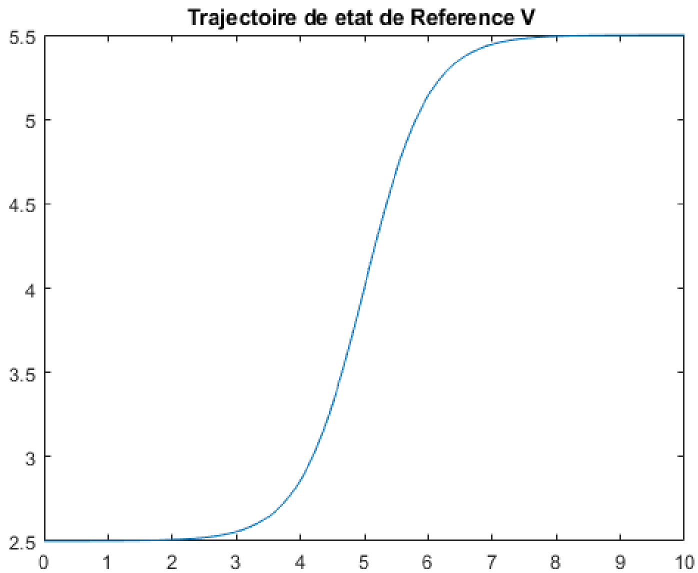

FIGURE 3 – Trajectoire de refeence pour l'état v

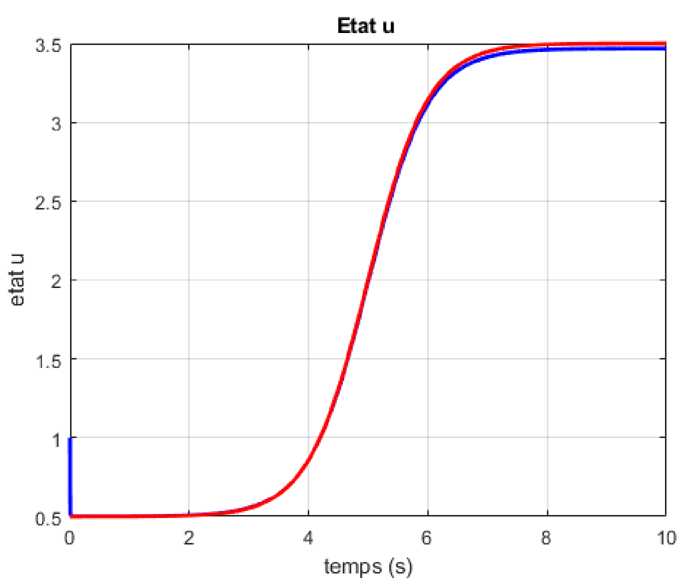

FIGURE 4 – suivie de trajectoire de pour l'état u

faite que nous avons initialise l'etat v_0 a 2. Puis, le trajectoire de l'etat arrive a suivre le trajectoire de reference d'une maniere effecase.

Conclusion partielle On voit bien qu'il y a un bon suivie de la trajectoire pour les deux état u et v.

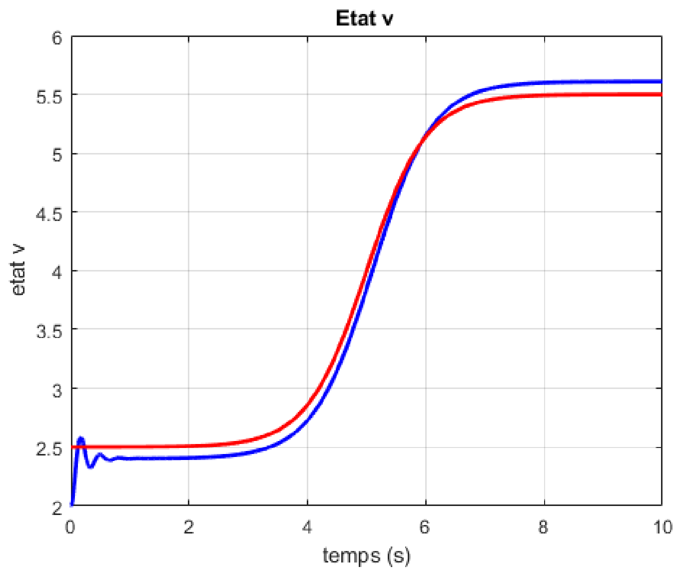

FIGURE 5 – suivie de trajectoire de pour l'état v

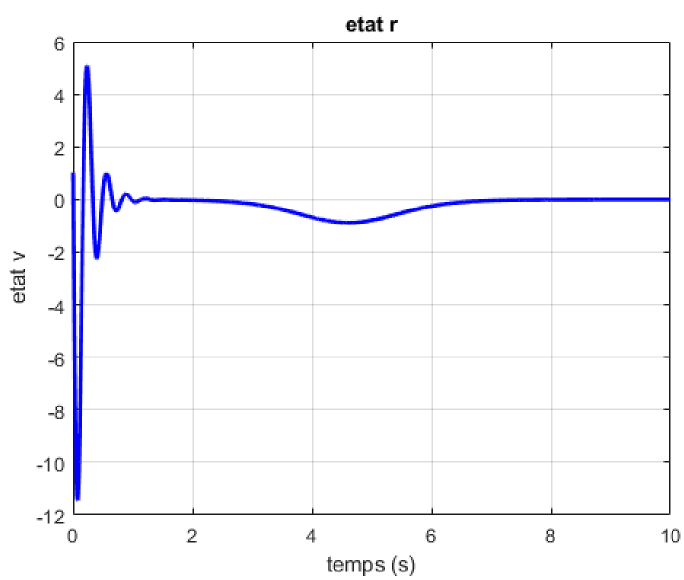

FIGURE 6 – l'evolution de l'état r

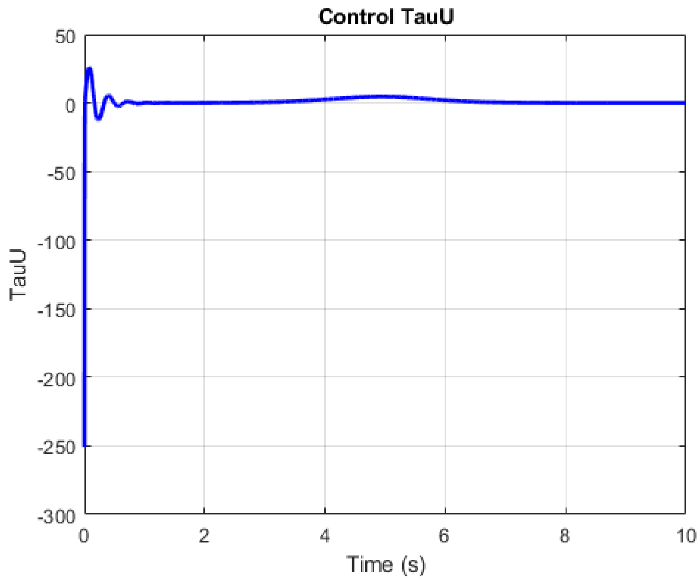

FIGURE 7 – commande τ_u

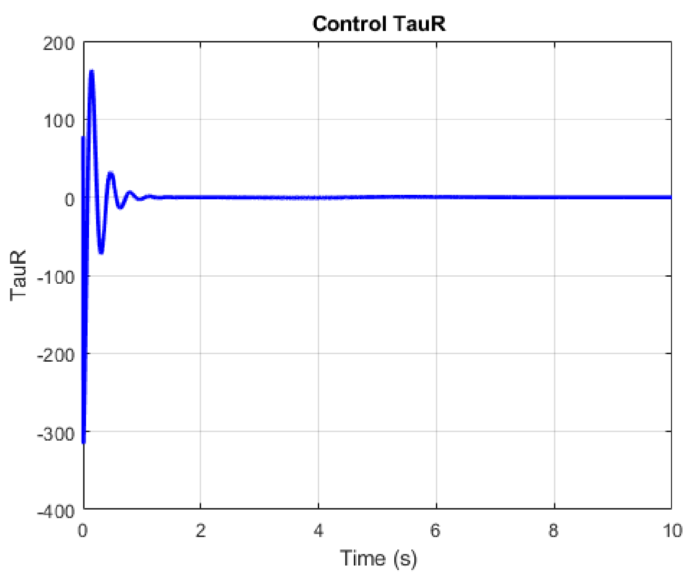

FIGURE 8 – commande τ_r

3.4 La performance du système sous perturbations

On n a essayé de simuler un suivie de trajectoire mais cette fois-ci en appliquant des perturbations sous la forme suivante :

On remarque les résultats suivantes :

Le trajectoire de l'etat u n'est pas affecte fortement mais si on fait zommer sur le debut du trajectoire, on peut voir des petites fluctuations. Pour l'etat v, on voit un effet claire sur le trajectoire sous forme des fluctuations ce qui donne une erreur plus important.

Conclusion partielle :

Apres avoir ajouter les perturbations on remarque que meme en presence de ces perturbations le systeme a reussit a corriger la commande de telle façon a suivre les trajectoire de reference pour les etat u et v.

4 Conclusion :

Le but de l'étude était principalement d'étudier la platitude du système et pour l'utiliser pour le suivi des trajectoires, nous avons utilisé l'algorithme d'extension dynamique pour valider la platitude de la sortie que nous avons choisie pour atteindre la trajectoire de rétroaction suivi. Nous avons implémenté le modèle et le suivi de trajectoire sur MatLab, ce qui a été fait avec succès on ajustant plusieurs paramètres. A la fin, on a testé la robustese on ajoutant une perturbation sur différents états du système et la simulation sur MatLab nous a montré des bonnes résultats.

Références

[1] Fantoni, I. et al. (1999). "Stabilization of a nonlinear underactuated hovercraft". In : Conference on Decision and Control, Phoenix, Arizona, USA, pp. 2533– 2538

[2] Sira–Ramirez, H. and C. Aguilar Ibanez (2000). "On the Control of the Hovercraft System". In : Dynamics and Control 10, pp. 151–163

[3] Morten Breivik, Vegard E. Hovstein, Thor I. Fossen. "Straight-Line Target Tracking for Unmanned Surface Vehicles", Modeling, Identification and Control, Vol. 29, No. 4, 2008, pp. 131—149

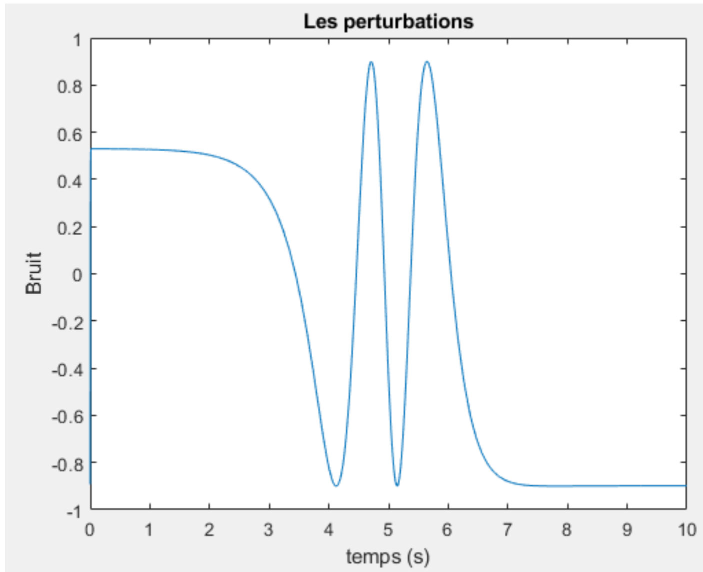

FIGURE 9 – Les perturbations

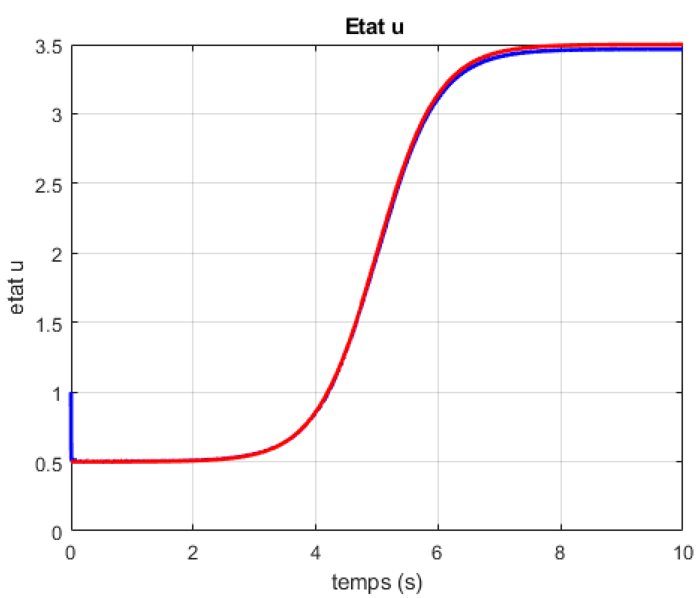

FIGURE 10 – suivie de trajectoire de pour l'état u

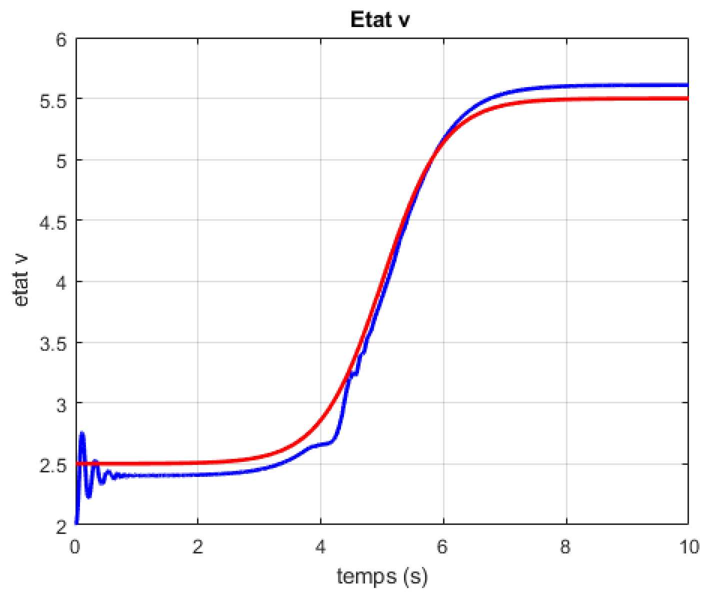

FIGURE 11 – l'evolution de l'état v

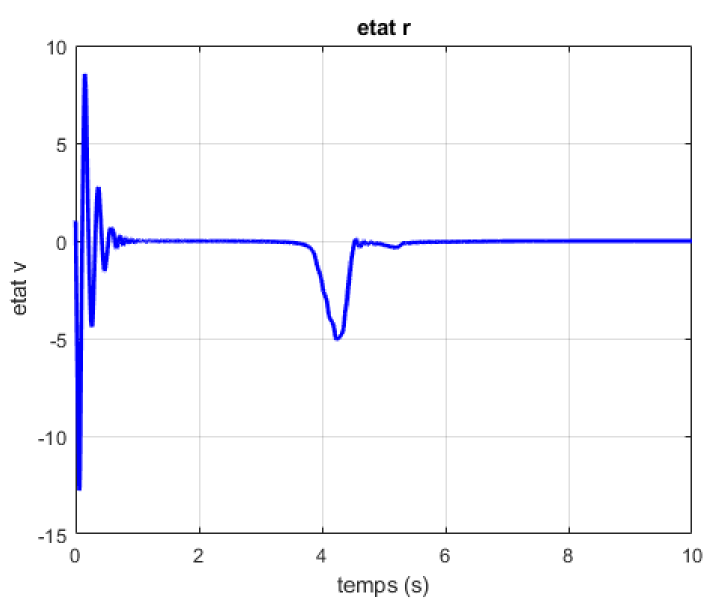

FIGURE 12 – l'evolution de l'état r

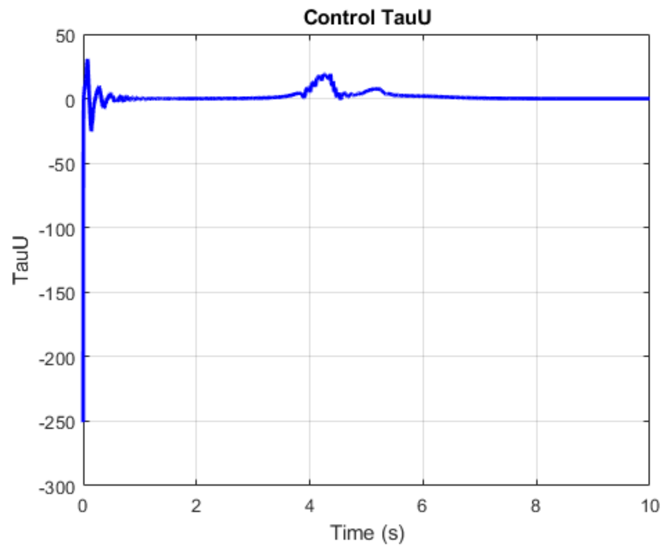

FIGURE 13 – commande τ_u

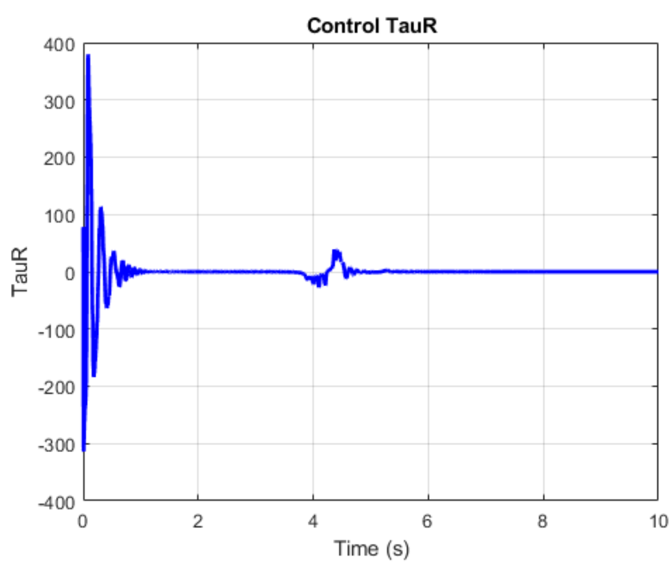

FIGURE 14 – commande τ_r