UNIVERSITÉ DU QUÉBEC EN ABITIBI-TÉMISCAMINGUE

MODÉLISATION ET SIMULATION D'UN QUADROTOR PAR COMMANDE BACKSTEPPING ADAPTATIVE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INGÉNIERIE

PAR

AMIR ZOUAOUI

OCTOBRE 2021

REMERCIMENTS

Avant d’entamer l’étude de ce mémoire, je veux remercier ceux qui m’ont donné l’occasion d’acquérir les connaissances professionnelles et d’approfondir certaines autres.

J’ai le plaisir de remercier M. Mohamad Saad, directeur de projet, qui m’a fait l’extrême honneur en me confiant un sujet si important. Je souhaite que ce travail soit à la hauteur de ce qu’il recherche, qu’il trouve ici le témoignage de mon profond respect.

Je tiens à remercier vivement M. Maarouf Saad, professeur à l’École de technologie supérieure (ÉTS), pour l’opportunité du stage à l’été 2019 au sein de son laboratoire de recherche à l’ETS, pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce à cette occasion, j'ai pu m'accomplir totalement dans mes missions.

Je n'oublie pas de remercier aussi les techniciens d’informatique de l’UQAT ainsi que mes collègues pour leurs soutiens et encouragements tout au long de la période du projet.

Enfin, j'adresse mes plus sincères remerciements à ma famille : mon père qui est décédé trop tôt, ma mère, mes deux frères et tous mes proches et tous mes amis en particulier Ahmed et Leanne, qui m'ont accompagné, aidé, soutenu et encouragé tout au long de la réalisation de ce mémoire.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

Tableau Page

RÉSUMÉ

Le grand intérêt porté aux robots volants a encouragé de nombreux travaux de recherche visant à améliorer les stratégies de contrôle. Ces travaux effectués sur ces robots volants ont montré que la configuration du quadrotor est la meilleure, grâce à la simplicité de sa structure, sa taille réduite, son faible poids, et son décollage et atterrissage vertical. Mais, malgré tous, ce dernier est un système non linéaire, multivariable, couplé et sous actionné. Donc, la commande nécessite une attention particulière, pour qu'elle soit une commande robuste et stable. Ce mémoire porte sur la modélisation, la conception et l'implémentation en temps réel d'un contrôleur par la commande backstepping adaptative sur la plateforme Parrot Mambo, pour valider la performance de la commande en termes de la stabilisation du quadrotor en vol stationnaire et le suivi de trajectoires désirées.

L'approche de la commande proposée est étudiée et appliquée au modèle Parrot Mambo pour contrôler le mouvement de rotation le long des axes x, y et z pour effectuer des vols stationnaires ainsi que le suivi d'une trajectoire désirée. Cette commande est basée sur le théorème de Lyapunov. Chaque état du système contrôle l'état précédent et est appelé "Commande virtuelle", jusqu'au dernier état qui est contrôlé par la commande réelle. L'idée est de calculer en plusieurs étapes une loi de commande qui assure la stabilité globale du système. Afin de rendre la commande robuste, une loi d'adaptation est associée à cette commande qui estime les paramètres inconnus et qui assure leur convergence vers leurs valeurs respectives, sans affecter le bon fonctionnement du système, notamment la stabilité.
La validation du contrôleur proposé a été effectuée en simulation, puis une implémentation expérimentale a été faite sur le matériel Parrot Mambo. Les résultats obtenus ont montré une performance en stabilisation en mode vol stationnaire ainsi que la poursuite d'une trajectoire désirée.

Mots-clés : quadrotor, commande non linéaire, backstepping, adaptative, stabilisation, suivi de trajectoire.

ABSTRACT

The great interest in flying robots has encouraged many research works to improve their control strategies. These works have shown that the quadrotor configuration is the best, thanks to the simplicity of its structure, its reduced size, its low weight, and its vertical take-off and landing. But, despite all, a flying robot is a non-linear, multivariable, coupled and underactuated system. Therefore, the control system needs special attention, so that it is a robust and stable control system. This thesis focuses on the modeling, design, and real-time implementation of a backstepping and adaptive control system on the Parrot Mambo platform to validate the performance of the control system in terms of stabilization of the quadrotor in hovering flight and tracking of the desired trajectory.

The proposed command is applied to the Parrot Mambo model to control the rotational motion along the x, y, and z axis to perform hovering and tracking of a desired trajectory. This control is based on the Lyapunov theory. Each state of the system controls the previous state and is called "virtual control", until the last state which is controlled by the real control. The idea is to compute in several steps a control law that ensures the global stability of the system. To make the control robust, an adaptation law is associated to this control that estimates the unknown parameters and ensures their convergence to their respective values, without affecting the proper functioning of the system, especially the stability.

The validation of the proposed controller was done in simulation, then an experimental implementation was done on the flying robot system to test closed loop performance.

Keywords quadrotor, non-linear control, backstepping, adaptive control, stabilization, trajectory tracking.

INTRODUCTION GÉNÉRALE

1. Introduction

L'industrie aéronautique constitue un secteur stratégique dans la dynamique du développement économique en influençant sur plusieurs secteurs dans le marché du transport civil, commercial et militaire. Il présente une forte croissance avec une technologie de pointe, notamment dans les aéronefs et les drones à commande numérique. Une attention particulière est accordée aux giravions, principalement, grâce à la variété de leurs applications possibles et de leurs maints avantages tel que : rapidité, précision, discrétion, réutilisation etc. La famille des véhicules aériens motorisés est subdivisée en plusieurs catégories. Drone ou quadrotor, équipé de quatre moteurs électriques et avec hélices mobiles, sont ceux qui ont gagné le plus en popularité. Les véhicules aériens sans pilote peuvent voler de manière autonome pour réaliser diverses applications telles que l'inspection, le contrôle et la surveillance dans les domaines de l'agriculture, l'exploitation minière et la construction. L'utilisation de ces robots volants a été d'abord connue dans les applications militaires, comme la surveillance et la reconnaissance. Ensuite, plusieurs applications civiles, dans lesquelles la présence de l'homme n'est pas indispensable, sont devenues concurrentes. Dans les missions actuelles, on peut citer la surveillance du trafic routier, l'inspection d'ouvrage, l'observation des phénomènes naturels, la pulvérisation des pesticides sur les surfaces agricoles, la surveillance de l'environnement par exemple : missions dangereuses, détection de gaz toxiques, radiations et la prévention des feux de forêt. Ils peuvent également être utilisés pour la photographie aérienne ou même pour la livraison tels que les médicaments d'urgence, des colis achetés sur le Net, des opérations de distribution des médicaments ou de la nourriture durant les catastrophes naturelles.

2. Motivation et mise en contexte

De nos jours, ces robots volants peuvent être déployés dans plusieurs domaines qui nécessitent un système moderne capable de fournir et transférer l'information. Les grandes entreprises du domaine multiplient leurs recherches dans le but de développer l'utilisation de tous les types de drone qui existent dans le marché et notamment les quadricoptères. Parrot, le leader européen de drones professionnels et civils, a fait un grand pas en avant par rapport à ses concurrents. Un excellent travail a été conçu, en faisant la famille des minidrones Parrot : Spider Rolling et Mambo. Le Parrot Mambo est un mini drone très agréable à piloter. Sa taille qui est relativement petite lui permet à la fois de survoler de manière contrôlée grâce à ses capteurs. N'étant pas tellement un drone standard, il ne peut pas être piloté avec des virages serrés. Il a été conçu, davantage, comme une démonstration technologique. En le comparant aux autres drones, et grâce à son autonomie, le vol du Parrot Mambo présente une solution efficace pour faire des tests, pour assurer la sécurité des gens et du matériel, ainsi que pour réduire les coûts d'opération.

De plus, MathWorks fournis plusieurs outils pour la gamme PARROT, qui peuvent nous servir pour la visualisation du comportement de vol et l'analyse des données de vol. En effet, les ingénieurs de Matlab ont développé le « Support Simulink for PARROT minidrones » pour les raisons suivantes : premièrement, aider les chercheurs à effectuer des expériences de contrôle en boucle fermée en classe ou en laboratoire, à partir des modèles, en utilisant des drones peu coûteux et de petite taille. Deuxièmement, aider les innovateurs et les chercheurs à comprendre et à adopter une conception basée sur un modèle à l'aide d'une solution éprouvée dans le domaine de l'enseignement supérieur. Et troisièmement, susciter l'intérêt et la conscience de l'impact de la conception à partir des modèles dans des applications critiques du monde réel. Parmi plus de 40 outils complémentaires Simulink, on peut citer les outils qu'on a utilisé dans ce travail : Aerospace Blockset, Simulink Coder et Simulink 3D animation.

3. Problématique

Comme la plupart des systèmes dynamiques non linéaires existants, la modélisation précise des robots volants et surtout les appareils télécommandés est difficile à obtenir. Les drones n'effectuent pas des trajets stables en raison de la forme des hélices qui sont ultra sensibles aux perturbations du vent et autres facteurs comme la géométrie de l'appareil. Les problèmes sus mentionnés réduisent la performance du système et affectent négativement le contrôle et le suivi de la trajectoire. La stabilité des quadricoptères et le suivi de la trajectoire dépendent du contrôle des quatre hélices. Six degrés de liberté sont contrôlés par quatre actionneurs. Pour cette raison, ce système est considéré à la fois comme un système sous-actionné et une structure dynamique fortement couplée. Ces défis de non-linéarité et de stabilité du système rendent la tâche du contrôle plus difficile. De plus, ils existent des utilisations complexes de ce type de drone, nécessitant des manœuvres agressives et, en même temps, la mise en œuvre d'un système de contrôle robuste. Les systèmes mécaniques du quadricoptère souffrent d'incertitudes. De tels problèmes peuvent être constatés au niveau du contrôle des drones surtout avec un changement de densité de l'air variant d'une altitude à une autre. L'aérodynamique et les caractéristiques de contrôle changent en fonction de l'altitude. Les effets environnementaux et le facteur de vieillissement jouent un rôle supplémentaire dans le changement des paramètres de la commande. Citons à titre d'exemple de perturbations météorologiques tels que la perturbation du vent et le frottement de l'air, et le couplage dynamique du modèle. Ce type particulier d'aéronef est relativement de petite taille et est considéré comme très sensible à la perturbation.

4. Objectif principale

L'objectif principal dans ce mémoire est d'appliquer une commande robuste permettant la commande de la position d'un robot volant de type quadrotor et permettant d'assurer sa stabilité, et de garantir la poursuite d'une trajectoire désirée.

Pour cela, on propose l'approche backstepping adaptative basée sur la théorie de Lyapunov qui convient bien aux systèmes sous-actionnés.

5. Méthodologie

Pour atteindre l'objectif principal, on va suivre la méthodologie suivante qui porte sur un flux de travail de simulation commun pour les drones et qui comprend les étapes suivantes :

- Le développement du modèle dynamique du robot volant du type quadrotor basé sur le formalisme de Newton-Euler.
- La conception d'une commande robuste Backstepping Adaptative pour la stabilisation en altitude, en attitude et en position.
- La validation de la performance de la loi de commande par simulation à l'aide de Matlab/Simulink en déterminant des gains liés à notre approche proposée pour assurer la meilleure stabilité et le bon suivie de la trajectoire désirée.
- La vérification et la validation du contrôleur avec un modèle aérodynamique réel basé sur « Aerospace Blockset » par la visualisation du comportement de vol.
- L'implémentation du contrôleur à l'aide de Matlab et la connexion Bluetooth pour la communication avec le Parrot Mambo.
- L'analyse et l'enregistrement des données de vol pour mettre en évidence la performance du système. La commande développée sur le modèle de quadrotor fournit de bons résultats en temps réel avec le minimum d'erreurs.

5.1. Structure du mémoire

Ce travail est organisé en cinq chapitres :

− Le premier chapitre, porte sur la revue de littérature. On va mettre l'accent sur les principales méthodes de contrôle des systèmes non linéaires qui ont été développées sur le modèle du quadrotor. Par la suite, la deuxième partie, porte sur la contribution du projet.

− Le deuxième chapitre, présente le modèle cinématique et dynamique du système en utilisant la méthode Newton-Euler. Une telle modélisation facilite la tâche pour trouver une solution et dégager la commande de contrôle la plus adéquate permettant la stabilisation du système.

− Le troisième chapitre, englobe la conception de la loi de commande robuste Backstepping adaptative qui se synthèse en deux étapes. La première étape, présente le développement de la loi de commande par la technique Backstepping en se basant sur la méthode de Lyapunov. Cette méthode a montré son efficacité pour assurer la stabilité du système, mais l'inconvénient majeur de cette approche est la difficulté à trouver une fonction appropriée candidate de Lyapunov ainsi que les problèmes de sensibilité à des paramètres incertains. Par la suite et en deuxième étape, une loi d'adaptation est associée à cette commande pour surmonter l'incertitude de ces paramètres et leurs changements qui se produisent dans le système dynamique.

− Le quatrième chapitre, présente une validation du contrôleur, conçue à l'aide du logiciel Matlab/Simulink, par les résultats de simulation et qui sera approuvé par une trajectoire complexe.

− Le cinquième chapitre, une implémentation du système a été faite avec l'outil « Simulink Support Package for Parrot Minidrones » qui assure la communication Bluetooth entre Matlab et le Parrot Mambo, pour tester et analyser les performances du système de commande. La commande développée sur le modèle de quadricoptère fournit de bons résultats en temps réel surtout en stabilisation du vol.

− En fin, on terminera par une conclusion générale et des recommandations.

CHAPITRE I REVUE DE LITTÉRATURE

1.1 Introduction

La dynamique du quadrotor est sous-actionnée, c'est un système non linéaire et hautement couplé. Les quatre entrées du drone, soient les quatre moteurs, contrôlent les six degrés de liberté (6-DOF), ce qui impose des contraintes sur le contrôle. Les incertitudes associées et la dynamique non linéaire exigent des approches de contrôle non linéaires. Au cours de la dernière décennie, la non-linéarité du modèle de quadricoptère UAV a suscité l'attention des chercheurs pour la conception des contrôleurs de vol non linéaires pour réaliser des manœuvres acrobatiques et des systèmes de commande autonomes. Un maintien de vol stable et un suivi de trajectoire désirée pour une enveloppe de vol plus long demeurent toujours un défi.

Les méthodes de contrôle non linéaires étudiées et appliquées au système de pilotage automatique du quadrotor comprennent la linéarisation par retour d'état, le contrôle Backstepping, le contrôle prédictif et la commande par mode glissant. Nous examinerons ainsi le contexte et la motivation de la mise en place de telles approches de commande.

1.2 Contexte et motivation

La conception d'un tel système nécessite une attention particulière, compte tenu de la dynamique non linéaire couplée, incertitudes paramétriques, problèmes de délai d'entrée, taux de convergence et problèmes de stabilité. La littérature sur les lois de commandes des systèmes non linéaires est vaste, très variée et peut mener à une meilleure compréhension sur les performances des contrôleurs proposés. Avant de passer à la discussion sur les techniques de commande non linéaire, il est inévitable de conférer la méthode de Lyapunov et d'autres notions de stabilité, parce que les termes

de stabilité jouent un rôle dans la conception de contrôle non linéaire et fournissent un point de référence pour une évaluation appropriée basée sur le rendement. D'après les définitions, on peut déduire que la notion la plus valable de stabilité est la stabilité exponentielle. Cependant, le critère rigoureux de la stabilité est la convergence en temps fini, qui assure une solution pour atteindre l'origine en temps fini. Après avoir décrit le mécanisme en général, on va mettre l'accent sur les principales méthodes de contrôle des systèmes non linéaires adoptées par la littérature.

1.3 Techniques de commande

Plusieurs techniques de contrôle ont été développées par différents groupes de recherche et universités dans le monde. Nous présentons quelques techniques utilisées dans la littérature :

Linéarisation par rétroaction :

Cette méthodologie de contrôle transforme la dynamique non linéaire du système en une dynamique linéaire équivalente, via une linéarisation par retour d'état ou de sorties. Nombreux chercheurs ont utilisé cette technique de contrôle pour concevoir un contrôleur de vol (Park, Kim et al. 2014). Comme la dynamique des robots volants de type quadrotor est non linéaire, cette approche est appropriée pour éliminer les non-linéarités, ainsi donc, le modèle du contrôle linéaire peut être utilisé pour concevoir un contrôleur de vol. Un inconvénient de la linéarisation par rétroaction est qu'elle ne vaut que pour une classe spécifique de systèmes non linéaires qui respectent la condition involutive Une étude spécifique sur la linéarisation exacte d'un quadricoptère a été présentée par Mistler et al. dans (Mistler, Benallegue et al. 2001). La linéarisation entrée-sortie du modèle quadricoptère, basée sur la représentation d'attitude quaternion, a été présentée dans (Wang and Jia 2014). La linéarisation des variables d'état x, y et z a été effectuée distinctement, ce qui aboutit à une linéarisation par rétroaction quasi-statique (Rudolph and Delaleau 1998). L'approche de linéarisation

par rétroaction, a été présentée par Lee et al (Lee, Kim et al. 2009). La commande de linéarisation par rétroaction adaptative a été conçue dans (Choi and Bang 2014) . Dans (Benallegue, Mokhtari et al. 2008), la linéarisation par rétroaction d'un quadricoptère a été réalisée avec la conception d'un observateur. Dans (Ghandour, Aberkane et al. 2014), un contrôleur de vol basé sur la linéarisation par rétroaction a été conçu pour un problème de suivi de la trajectoire ainsi qu'une discussion d'une solution en cas de défaillance du rotor. Une étude similaire a été menée dans (Zhou, Zhang et al. 2010).

La commande par mode glissant :

La commande par mode glissant, est une technique avancée de contrôle non linéaire. Cette méthode consiste à modifier la dynamique d'un système non linéaire en lui appliquant un signal de commutation haute fréquence le forçant à rejoindre, puis à rester sur une surface de glissement. Cette loi de contrôle est une approche jugée robuste aux perturbations externes et aux incertitudes paramétriques. Cette propriété a été démontrée par plusieurs chercheurs tels que C'est ainsi que cette architecture de commande fût implantée sur les hélicoptères (McGeoch, McGookin et al. 2005) et les quadrotors. (Xu and Ozguner 2006) ont proposé une commande par mode glissant avec une commande par PID pour le contrôle et la stabilisation d'un hélicoptère de type quadrotor. (Bouadi, Bouchoucha et al. 2007) ont développé une commande par mode glissant pour garantir le suivi d'une trajectoire désirée et assurer la stabilité au sens de Lyapunov (Besnard, Shtessel et al. 2012) et (Sumantri, Uchiyama et al. 2013)

Commande basée sur la théorie de Lyapunov :

Cette technique a permis de démontrer que le quadrotor est asymptotiquement stable sous certaines conditions (Bouabdallah 2007), (Castillo, Dzul et al. 2004), et (Castillo, Lozano et al. 2004).

La commande LQR :

Cette loi de commande a donné de bons résultats dans la stabilisation d'attitude du quadrotor OS4 dans les travaux de S. Bouabdallah et André Noth (Bouabdallah, Noth et al. 2004). Ces résultats ont été comparés avec ceux obtenus par un contrôleur PID.

La commande « Dynamic Feedback » :

Cette technique a été appliquée dans quelques projets sur les quadrotors. L'objectif est de transformer le système en boucle fermée en sous-systèmes linéaires, contrôlables et découplés (Bresciani 2008), (Mokhtari and Benallegue 2004).

La Commande par Vision :

Cette technique est basée sur la commande visuelle utilisant soit une caméra miniature embarquée à bord du quadrotor, ou une caméra externe (Altug 2003).

L'approche Backstepping :

L'approche de contrôle par backstepping est basée sur la théorie de Lyapunov et convient bien aux systèmes sous-actionnés. C'est un schéma récursif de la conception d'un contrôleur, dans lequel la procédure de la conception de la commande commence à partir d'un sous-système stable bien déterminé et chaque nouveau contrôleur converge progressivement dans les sous-systèmes externes. Cette progression dure jusqu'à l'avènement du terme de contrôle principal. C'est la principale caractéristique de l'approche par backstepping. L'entrée de commande est choisie par une autre méthode basée sur la fonction de Lyapunov et connue pour ce sous-système stable. La conception du contrôle backstepping offre une méthode qui étale une stabilité contrôlée de ce sous-système au système principal. Plusieurs études ont appliqué cette approche dans le but de concevoir un contrôleur de vol pour un quadricoptère. Dans (Dolatabadi and Yazdanpanah 2015), le backstepping, basé sur un contrôleur non linéaire, a été conçu pour la stabilisation de la boucle interne d'orientation tandis que la boucle externe de position était contrôlée à l'aide d'un contrôleur linéaire. Dans (Yali and

Yuanxi 2012), le contrôle de suivi d'altitude a été effectué à l'aide d'un contrôleur backstepping. Une autre étude approfondie sur la conception du contrôle par backstepping a été présentée par (Madani and Benallegue 2006). L'ensemble du système a été divisé en trois sous-systèmes : un sous-système entièrement actionné, le deuxième est sous-actionné et un troisième propulsé. Toutes les variables d'état du système étaient contrôlées à l'aide de la technique de contrôle susmentionnée et la convergence était garantie par la théorie de Lyapunov. Une version plus étendue de cette étude était présentée dans (Madani and Benallegue 2007), où une entrée de commande virtuelle du modèle de backstepping était basée sur un contrôleur en de mode glissant de deuxième ordre. Cela réduit la loi de contrôle en éliminant le besoin de considérer les dérivées de la dynamique du système. Une autre contribution, lors de la conception du contrôleur par backstepping, faite dans (Das, Lewis et al. 2009), où le contrôle de backstepping a été appliqué sur la forme de dynamique Lagrangienne de retour d'état. Dans (Bouabdallah and Siegwart 2005), le contrôleur de backstepping est conçu et a montré la capacité de contrôler les angles d'attitude en présence des perturbations. Les auteurs ont conclu que, si les termes d'intégrale ont été ajoutés dans le backstepping, l'approche de contrôle résultante devient plus robuste et l'erreur en régime permanent est éliminée. Ils ont conçu un contrôleur de vol intégral basé sur la backstepping pour le contrôle de suivi du quadricoptère.

Une étude a été menée dans (Mian, Ahmad et al. 2008), où les auteurs ont inséré un terme intégral au début de la conception de backstepping. Ceci a finalement abouti à un contrôleur non linéaire de type PID, ayant trois paramètres de contrôle réglables. La conception entière était basée sur les fonctions de Lyapunov. La combinaison du PID et du backstepping est une bonne idée pour concevoir un contrôleur qui montre relativement plus de robustesse vis-à-vis les perturbations externes, en le comparant au contrôleur de backstepping conventionnel. Une étude très récente (Du, Zhu et al. 2017) a abordé la convergence en temps fini des erreurs vers l'origine en utilisant la commande backstepping pour le contrôle de formation des quadrotors. La convergence

en temps fini a été obtenue en utilisant la théorie des systèmes homogènes et la théorie de la stabilité de Lyapunov. Dans (Bouabdallah and Siegwart 2005), cette commande a été implémentée avec une action intégrale pour permettre d'éliminer l'erreur en régime permanent tout en garantissant une stabilité asymptotique. Les auteurs dans (Bouchoucha, Tadjine et al. 2008), ils ont présenté une stratégie de contrôle backstepping basée sur le modèle avec le schéma de réglage de gain proportionnel.

Une autre étude présentée dans (Yu, Guo et al. 2015), contient la technique de compensation du signal avec la conception de contrôleur de backstepping. Cette approche a été utilisée pour éliminer l'inconvénient de l'explosion du calcul qui est inhérent à la conception de la commande backstepping. Dans (Hamel, Mahony et al. 2002), les auteurs ont également proposé un contrôleur, non linéaire, basé sur le backstepping, pour stabiliser le quadricoptère. La méthodologie, qui a été mise en œuvre dans leur travail, sépare la dynamique du corps rigide de la dynamique du rotor et englobe ensuite les fonctions candidates de Lyapunov séparées pour la dynamique du quadricoptère. Les erreurs générées, séparément, utilisent l'argument de délimitation transitoire pour entrer dans une seule fonction candidate de Lyapunov. Par conséquent, cette approche garantit une stabilité pratique améliorée pour le modèle du quadricoptère non linéaire. De plus, elle évite la réquisition de la linéarisation approximative et l'extension dynamique pour la conception du contrôleur. La représentation quaternion a été utilisée pour décrire la rotation des angles d'attitude qui permet de générer un contrôle statique en douceur pour la stabilisation de l'attitude. La stabilité asymptotique est garantie par la théorie de Lyapunov. Il existe de nombreux problèmes associés à la non-linéarité dynamique du quadricoptère. Citons, à titre d'exemple, l'incertitude du tenseur d'inertie et masse du corps, les retards d'entrée, les perturbations externes inconnues, les dérives des capteurs, etc. L'incertitude de masse et d'inertie provient du fait que différents types de missions d'UAV peuvent avoir des charges utiles différentes. Cela étant, pour compenser ces incertitudes, plusieurs études ont proposé des lois d'adaptation différentes. Une autre étude inclusive sur la

commande backstepping adaptative a été proposée. La technique de contrôle de backstepping a été présentée dans (Fang and Gao 2011). Dans une étude récente (Mohd Basri, Husain et al. 2015), les perturbations incertaines ont été, en approximation, en ligne à l'utilisation des fonctions de base radiale et un contrôleur backstepping adaptatif a été conçu et fonctionnait bien sous des perturbations inconnues. Très peu d'études expérimentales existent dans la littérature sur la commande backstepping adaptative pour la commande de vol du quadricoptère. De plus, en ce qui concerne le taux de convergence, sont peu les articles qui ont évoqué la question des convergences temporelles des erreurs de suivi de la trajectoire. Les contrôleurs de vol backstepping ont encore besoin de l'attention des chercheurs pour être conçus en tenant compte d'une stabilité plus rigoureuse et de taux de convergence plus rapide.

La commande adaptative :

L'approche adaptative réduit l'erreur de suivi et améliore les performances du contrôleur. L'adaptation permet la modification des paramètres de contrôle, ce qui donne des performances supérieures du système de commande , notamment avec les dynamiques non modélisables et l'incertitude des paramètres (Bresciani 2008). Dans le travail de (Amieur et Boumehraz, 2014), une commande adaptative a été combiné avec mode glissement pour assurer la robustesse et atténuer les effets des perturbations externes pour un système non linéaire. Un autre travail de (El Kari, Essounbouli et Hamzaoui, 2003) ont proposé quant à eux une commande adaptative floue robuste dont le but de résoudre le problème de poursuite d'un système non linéaire incertain avec des perturbations externes. Dans, le travail de (Shaiful et Hazry, 2011) ont utilisé une commande adaptative à base de réseau de neurones pour le contrôle d'attitude et la stabilisation du quadrotor. (Bouadi, Bouchoucha et Tadjine, 2007b) ont développé une commande par mode glissant basée sur une commande par backstepping adaptative dont le but de minimiser l'erreur de suivi et d'assurer la stabilité de Lyapunov. Une commande de linéarisation par rétroaction adaptative a été conçue dans (Choi and Bang

2014). L'approche adaptative réduit l'erreur de suivi et améliore les performances du contrôleur.

1.4 Contribution

Le suivi de la trajectoire et la stabilité du quadrotor dépendent du contrôle des quatre moteurs. Le mouvement du quadrotor est contrôlé par la variation de vitesse de chaque rotor pour changer la force de portance et le couple crée par chacun. Les six degrés de liberté sont contrôlés par quatre entrées. Donc, c'est clair que ce système est sous actionné vue que le nombre des actionneurs est inférieure au nombre des degrés de liberté. Ce qui rend la tâche du contrôle plus difficile et la nécessité d'un contrôleur robuste face aux changements de paramètres qui se produisent dans le système dynamique.

Il en résulte, de ce qui précède, que ces robots volants, en tant que système robotique, souffre de non-linéarités dures, d'une dynamique non modélisée et des perturbations externes. En majorité, les robots sont des systèmes dynamiques non linéaires. Les contrôleurs non linéaires ont montré leurs performances pour assurer leur stabilité. De nombreuses recherches ont été menées sur cette question dans le but d'améliorer la qualité du contrôle non linéaire et d'éviter tout défaut éventuel. Prenons à titre d'exemple, la commande par backstepping est l'un des systèmes de contrôle non linéaire le plus efficace.

On va présenter les concepts de base de la commande par backstepping qui est basée sur le théorème de Lyapunov. Chaque état du système commande l'état précédent et est appelé "Commande virtuelle", jusqu'au dernier état qui est commandé par la commande réelle. L'objectif de cette technique est de calculer en plusieurs étapes une loi de commande qui assure la stabilité globale du système.

Ensuite, une loi d'adaptation est associée à cette commande qui estime les paramètres inconnus et qui assure leur convergence vers leurs valeurs respectives.

On peut résumer les contributions sur les points suivants :

- Utilisation de l'approche Backstepping pour assurer la stabilité du système. La convergence des variables d'état internes du quadrotor a été garantie, quels que soient les états initiaux.
- Estimation des paramètres du système par une loi d'adaptation associée à cette commande Backstepping pour la robustesse de la commande, afin d'assurer la convergence vers leurs valeurs respectives, sans affecter le bon fonctionnement du système, notamment la stabilité.
- Validation du de la performance de la loi commande robuste Backstepping-Adaptative pour la stabilité d'un vol stationnaire et le suivie d'une trajectoire désirée par deux étapes : la première est par Simulink et la deuxième par l'implémentation du notre contrôleur sur le matériel Parrot Mambo.

CHAPITRE II MODÉLISATION DU QUADROTOR

2.1 Introduction

Nul ne peut nier qu'une bonne modélisation du système est essentielle pour la réalisation de son contrôle. Une telle modélisation par Newton-Euler, sera élaboré dans ce chapitre. Tout d'abord, il faut bien comprendre les mouvements que peut effectuer ce robot volant et établir son modèle cinématique et dynamique, vue que ce système est complexe, non linéaire et multivariable. Cette tâche, nous permettra par la suite dans les chapitres suivants de trouver plus facilement une solution pour commander le système.

2.2 Modèle mathématique du quadrotor

Le modèle dynamique du quadrator est donné par le formalisme de Newton-Euler

2.2.1 Hypothèses du modèle

Pour pouvoir déterminer le modèle dynamique, on suppose que :

- Le quadrotor a une structure rigide et symétrique d'où l'hypothèse que la matrice d'inertie est diagonale.
- Les hélices sont également supposées rigides pour pouvoir négliger l'effet de leur déformation lors de la rotation.
- La trainée de chaque moteur est proportionnelle au carré de la vitesse de rotation des rotors.
- Le repère lié à cette structure est confondu avec son centre de gravité.

Donc on peut considérer que la dynamique du quadrotor comme un corps solide dans l'espace.

2.2.2 Définition des repères

L'étude du mouvement d'un quadrotor évoluant dans l'espace, nécessite la connaissance de sa position et de son orientation. Donc cette localisation nécessite le choix d'au moins deux repères :

- **Le repère terrestre :** On le note $R_0\ (X_0, Y_0, Z_0)$.
- **Le repère local** $R_G\ (X, Y, Z)$ est attaché au centre de masse G du véhicule

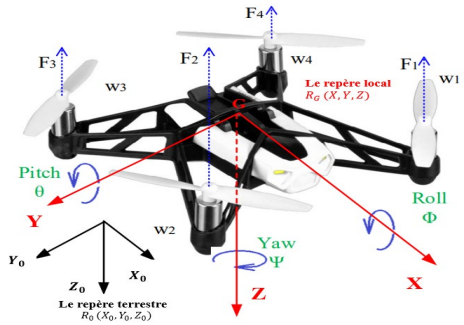

Figure 2.1 Repérage du Quadrotor

2.2.3 Angles d'Euler

Le quadrotor est un robot volant à six degrés de liberté, soient trois translations (x, y, z) et trois rotations (ϕ, θ, ψ).

Les rotations du quadrotor sont décrites par trois rotations consécutives à travers trois angles qui s'appellent les angles d'Euler.

Le passage du repère R_0 vers le repère R_G se fait par trois rotations en utilisant deux repères intermédiaires R_i et R_j.

On fait une rotation autour de l'axe $X_i = X_0$. On passe du repère R_0 vers R_i en faisant une rotation d'angle ϕ appelé angle de roulis. On obtient alors Rot(x, ϕ) qui est une matrice de rotation orthogonale d'angle ϕ autour de O_x , représentée par :

$$Rot(x, \emptyset) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\emptyset & -sin\,\emptyset \\ 0 & sin\,\emptyset & cos\,\emptyset \end{bmatrix} \quad (2.1)$$

❖ **Passage du repère R_i vers R_j**

On fait une rotation autour de l'axe Y_0 . On passe du repère R_i vers R_j en faisant une rotation d'angle θ appelé angle de tangage. On obtient alors Rot (y, θ) qui est la matrice de rotation orthogonale d'angle θ autour de O_y représentée par :

$$Rot(y, \theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2.2)$$

❖ **Passage du repère R_j vers R_G**

On fait une rotation autour de l'axe z_0 On passe du repère R_j vers R_G en faisant une rotation d'angle ψ appelé angle de lacet. On obtient alors Rot(z, ψ) : est la matrice de rotation orthogonale d'angle ψ autour de O_z, et qui est représentée par :

$$Rot(z, \psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

La matrice de passage du repère R_0 vers le repère R_G est donné par le produit des trois matrices successives :

$$Rot_{\Psi\theta\emptyset} = Rot(z,\Psi) * Rot(y,\theta) * Rot(x,\emptyset) \tag{2.4}$$

Après développement, et avec la notation (c : cos, s : sin), on obtient donc :

$$Rot_{\Psi\theta\emptyset} = \begin{bmatrix} \mathrm{c}\,\theta\,\mathrm{c}\,\Psi & s\phi\,\mathrm{c}\,\Psi\,s\,\theta - s\psi\,c\phi & \mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset \\ \mathrm{c}\,\theta\,\mathrm{s}\,\Psi & s\theta\,s\psi\,s\phi + c\psi\,c\phi & \mathrm{c}\,\emptyset\,\mathrm{s}\,\theta\,\mathrm{s}\,\Psi - c\psi\,s\phi \\ -s\theta & \mathrm{c}\,\theta\;\;s\phi & \mathrm{c}\,\theta\,\mathrm{c}\,\phi \end{bmatrix} \tag{2.5}$$

2.2.4 Modélisation avec le formalisme de Newton-Euler

2.2.4.1 Dynamique de translation

D'après la deuxième loi de la dynamique de Newton :

$$\frac{d(mV)}{dt} = \sum F_{ext \rightarrow repère} \tag{2.6}$$

La vitesse exprimée dans le repère inertiel est :

$$\frac{d(mV)}{dt} = m\dot{V} \tag{2.7}$$

Avec : V = (ẋ, ẏ, ż) : Vecteur de vitesses de translation de la plateforme exprimée dans le repère R_0 , m : La masse du corps.

Les forces extérieures appliquées sur le quadrotor sont :

- Force de gravité

$$F_g = m * g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

Avec : m : La masse du corps, g : L'accélération de la gravité, $F_{portance}$: Elle est perpendiculaire à l'écoulement d'air, et est dirigée vers le haut c'est-à-dire qu'elle a tendance à faire élever le quadrirotor. Cette force représente la force totale produite par les quatre hélices, elle est représentée dans le repère terrestre R_0 (X_0, Y_0, Z_0), et donnée par l'expressions suivante :

$$F_{Portance} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} Rot_{\Psi\theta\emptyset} \begin{bmatrix} 0 \\ 0 \\ F_p \end{bmatrix} \tag{2.9}$$

$$F_{portance} = \begin{bmatrix} (\mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset)\,F_p \\ (\mathrm{c}\,\emptyset\,\mathrm{s}\,\theta\,\mathrm{s}\,\Psi - c\psi\,s\phi)\,F_p \\ -(\mathrm{c}\,\theta\,\mathrm{c}\,\phi)\,F_p \end{bmatrix} \tag{2.10}$$

Avec :

$$F_p = \sum_{i=1}^{4} f_i \tag{2.11}$$

✓ F_p : la poussée totale, f_i : est la force produite par la rotation de l'hélice *i*. Cette force est proportionnelle au carré de la vitesse de rotation des moteurs et, elle est donnée par :

$$f_i = b * {\omega_i}^2 \tag{2.12}$$

Avec : b : est le coefficient de poussée qui dépend des propriétés aérodynamiques, ω_i: est la vitesse de rotation du moteur *i.*

- Les forces de trainée :

C'est la résultante des forces qui s'opposent au mouvement du quadrirotor dans l'air, de même direction que le mouvement du quadrotor mais de sens opposé. Elle représente en quelque sorte les forces de frottement visqueux sur l'objet. Ces forces sont données par :

$$F_t = -\begin{bmatrix} k_{ftx} & 0 & 0 \\ 0 & k_{fty} & 0 \\ 0 & 0 & k_{ftz} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.13}$$

Avec : k : coefficient de trainée (drag)

L'équation (2.6) s'écrit sous la forme :

$$m\dot{V} = F_{portance} + F_t + F_g \tag{2.14}$$

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (\mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset)F_p \\ (\mathrm{c}\,\emptyset\,\mathrm{s}\,\theta\,\mathrm{s}\,\Psi - c\psi\, s\phi)\, F_p \\ -(\mathrm{c}\,\theta\,\mathrm{c}\,\phi)F_p \end{bmatrix} - \begin{bmatrix} k_{ftx}\,\dot{x} \\ k_{fty}\,\dot{y} \\ k_{ftz}\,\dot{z} \end{bmatrix} + m * g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.15}$$

On obtient donc :

$$\begin{cases} \ddot{x} = \dfrac{1}{m}\ \left[(\mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset)\, F_p\right] - k_{ftx}\,\dot{x} \\ \ddot{y} = \dfrac{1}{m}\ \left[(\mathrm{s}\,\theta\,\mathrm{s}\,\Psi\,\mathrm{c}\,\emptyset - \mathrm{c}\,\Psi\,\mathrm{s}\,\emptyset)\, F_p\right] - k_{fty}\,\dot{y} \\ \ddot{z} = g - \dfrac{F_p}{m}\ (\mathrm{c}\,\theta\ \mathrm{c}\,\phi) - \dfrac{k_{ftz}}{m}\dot{z} \end{cases} \tag{2.16}$$

2.2.4.2 . Dynamique de rotation

D'après la deuxième loi de la dynamique de Newton :

$$\frac{d(J\Omega)}{dt} = \sum M_{rep \rightarrow repère} \tag{2.17}$$

Comme la vitesse angulaire est exprimée dans le repère lié au quadrotor, alors :

$$\frac{d(J\Omega)}{dt} = J\dot{\Omega} + \Omega \text{ x J } \Omega \tag{2.18}$$

Avec : $\Omega = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$.

Ω : est le vecteur de vitesses de rotation du quadrotor exprimé dans le repère R_G (lié à la plateforme) défini en fonction des variations des angles roulis, tangage et lacet (ϕ, θ, Ψ) et décrites dans le repère inertiel, J : est la matrice d'inertie. Comme la structure du quadrirotor est supposée symétrique, la matrice d'inertie J est diagonale :

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.19}$$

Les moments sont :

- Le moment causé par les forces de poussée et de trainée

$$M_f = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{2.20}$$

Les deux moments $M_x$et M_y : sont les responsables à la rotation autour des axes x et y. M_x est dû à la différence entre les forces de portances des rotors (3,4) et les rotors (1,2)

dans la direction x. M_y est dû à la différence entre les forces de portances des rotors (2,3) et les rotors (1,4) dans la direction y.

Ces mouvements sont donnés par les expressions suivantes :

$$M_x = bl\,(\omega_1{}^2 - \omega_2{}^2 - \omega_3{}^2 + \omega_4{}^2) \qquad (2.21)$$
$$M_y = bl\,(\omega_1{}^2 + \omega_2{}^2 - \omega_3{}^2 - \omega_4{}^2)$$

M_z est dû à la différence de vitesse dans les deux couples d'hélices autour de l'axe z. Ce moment est donné par l'expression suivante :

$$M_z = d\,(-\omega_1{}^2 + \omega_2{}^2 - \omega_3{}^2 + \omega_4{}^2) \qquad (2.22)$$

$$M_f = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} l * b * (\omega_1{}^2 - \omega_2{}^2 - \omega_3{}^2 + \omega_4{}^2) \\ l * b * (\omega_1{}^2 + \omega_2{}^2 - \omega_3{}^2 - \omega_4{}^2) \\ d * (\omega_2{}^2 + \omega_4{}^2 - \omega_1{}^2 - \omega_3{}^2) \end{bmatrix} \qquad (2.23)$$

Avec : l : la distance entre le centre du quadrotor et axe de rotation du rotor, d : le coefficient aérodynamique de trainée dépendant des propriétés aérodynamiques, ω: la vitesse angulaire dans le repère fixe.

- Le moment résultant du frottement aérodynamique, il est donné par :

$$M_a = \begin{bmatrix} k_{fax}\,\dot{\phi}^2 \\ k_{fay}\,\dot{\theta}^2 \\ k_{faz}\,\dot{\psi}^2 \end{bmatrix} \qquad (2.24)$$

Avec : k_{fax} , k_{fay}, et k_{faz}: sont les coefficients aérodynamiques.

- Le moment gyroscopique :

Le moment gyroscopique est dû à l'inertie des rotors J_r et à la vitesse ω_r. L'expression générale de ce moment est donnée par :

$$M_g = \begin{bmatrix} J_r \ \dot{\theta} \ \omega_{\mathrm{r}} \\ -J_r \ \dot{\phi} \ \omega_{\mathrm{r}} \\ 0 \end{bmatrix} \tag{2.25}$$

Avec J_r : la matrice d'inertie du rotor, ω_r : est la vitesse de rotation du quadrotor dans le repère local .

Avec

$$\omega_{\mathrm{r}} = \ -\omega_1 + \omega_2 - \omega_3 + \omega_4$$

$$J_r = \begin{bmatrix} J_{rx} & 0 & 0 \\ 0 & J_{ry} & 0 \\ 0 & 0 & J_{rz} \end{bmatrix} \tag{2.26}$$

L'équation (2.18) devient :

$$J\dot{\Omega} + \Omega \ \mathrm{X} \ \mathrm{J} \ \Omega = -M_a - M_g + M_f \tag{2.27}$$

On calcule le terme Ω X (JΩ) :

$$\Omega \ \mathrm{X} \ (J \ \Omega) = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \mathrm{X} \left(\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) \tag{2.28}$$

$$\Omega \ \mathrm{X} \ (J \ \Omega) = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \mathrm{X} \ \begin{bmatrix} I_{xx} \ \dot{\phi} \\ I_{yy} \ \dot{\theta} \\ I_{zz} \ \dot{\psi} \end{bmatrix} \tag{2.29}$$

$$\Omega \text{ X } (J\,\Omega) = \begin{bmatrix} (I_{zz} - I_{yy})\ \dot{\theta}\,\dot{\psi} \\ (I_{xx} - I_{zz})\,\dot{\phi}\,\dot{\psi} \\ (I_{yy} - I_{xx})\,\dot{\phi}\,\dot{\theta} \end{bmatrix} \tag{2.30}$$

On calcule le terme $J\,\dot{\Omega}$:

$$J\,\dot{\Omega} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \tag{2.31}$$

$$J\,\dot{\Omega} = \begin{bmatrix} I_{xx}\,\ddot{\phi} \\ I_{yy}\,\ddot{\theta} \\ I_{zz}\,\ddot{\psi} \end{bmatrix} \tag{2.32}$$

Remplaçant les termes (2.32), (2.30), (2.24), (2.23) et (2.25) dans l'équation (2.27) :

$$\begin{bmatrix} I_{xx}\,\ddot{\phi} \\ I_{yy}\,\ddot{\theta} \\ I_{zz}\,\ddot{\psi} \end{bmatrix} + \begin{bmatrix} (I_{zz} - I_{yy})\,\theta\,\psi \\ (I_{xx} - I_{zz})\,\phi\,\psi \\ (I_{yy} - I_{xx})\,\phi\,\psi \end{bmatrix} = -\begin{bmatrix} k_{fax}\,\dot{\phi}^2 \\ k_{fay}\,\dot{\theta}^2 \\ k_{faz}\,\dot{\psi}^2 \end{bmatrix} - \begin{bmatrix} J_r\ \dot{\theta}\,\omega_{\mathrm{r}} \\ -J_r\ \dot{\phi}\,\omega_{\mathrm{r}} \\ 0 \end{bmatrix} + \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{2.33}$$

$$\begin{cases} \ddot{\phi} = \dfrac{1}{I_{xx}}\ \left[(I_{yy} - I_{zz})\,\dot{\theta}\ \dot{\psi} - k_{fax}\,\dot{\phi}^2 - J_r\,\dot{\theta}\,\omega_{\mathrm{r}} + M_x\right] \\ \ddot{\theta} = \dfrac{1}{I_{yy}}\ \left[(I_{zz} - I_{xx})\,\dot{\phi}\,\dot{\psi} - k_{fay}\,\dot{\theta}^2 + J_r\,\dot{\phi}\,\omega_{\mathrm{r}} + M_y\right] \\ \ddot{\psi} = \dfrac{1}{I_{zz}}\ \left[(I_{xx} - I_{yy})\,\dot{\phi}\,\dot{\theta}\ - k_{faz}\,\dot{\psi}^2 + M_z\right] \end{cases} \tag{2.34}$$

2.3 Matrice de transformation entre force / moments et vitesses des moteurs :

Après avoir présenté l'ensemble des forces et des moments qui agit sur le système, la matrice suivante relie ces grandeurs avec les vitesses de rotation des moteurs. Cette relation est très importante pour l'implémentation du contrôleur.

Les entrées du système sont $(U_1, U_2, U_3 et\ U_4)$, les vitesses des moteurs ω_{i} et ω_{r} la vitesse de quadrator :

$$\begin{cases} U_1 = b\ ({\omega_1}^2 + {\omega_2}^2 + {\omega_3}^2 + {\omega_4}^2) \\ U_2 = bl\ ({\omega_1}^2 - {\omega_2}^2 - {\omega_3}^2 + {\omega_4}^2) \\ U_3 = bl\ ({\omega_1}^2 + {\omega_2}^2 - {\omega_3}^2 - {\omega_4}^2) \\ U_4 = d\ (-{\omega_1}^2 + {\omega_2}^2 - {\omega_3}^2 + {\omega_4}^2) \end{cases}$$

$$U = Q\ f$$

$$U = \begin{bmatrix} F_p \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ bl & -bl & -bl & bl \\ bl & bl & -bl & -bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \end{bmatrix} \tag{2.35}$$

L'inverse de la matrice carrée de l'équation (2.35) donne les expressions des vitesses des moteurs.

$$f = Q^{-1}\ U \tag{2.36}$$

On obtient donc l'équation suivante ;

$$\begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \end{bmatrix}_{Calculé} = \begin{bmatrix} \frac{1}{4b} & \frac{1}{2(2bl)} & \frac{1}{2(2bl)} & \frac{-1}{4d} \\ \frac{1}{4b} & \frac{-1}{2(2bl)} & \frac{1}{2(2bl)} & \frac{1}{4d} \\ \frac{1}{4b} & \frac{-1}{2(2bl)} & \frac{-1}{2(2bl)} & \frac{-1}{4d} \\ \frac{1}{4b} & \frac{1}{2(2bl)} & \frac{-1}{2(2bl)} & \frac{1}{4d} \end{bmatrix} \begin{bmatrix} F_p \\ M_x \\ M_y \\ M_z \end{bmatrix} \tag{2.37}$$

2.4 Représentation d'état

Les équations (2.16) et (2.34) donnent le modèle dynamique complet du quadrotor. Ceci est en accord avec les travaux présentés dans (Alqaisi 2019) et (Alqaisi, Brahmi et al. 2019).

$$\left\{ \begin{array}{l|l} \ddot{x} = \frac{1}{m}\ [(\mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset)\,F_p] & \ddot{\phi} = \frac{1}{I_{xx}}\ [(I_{yy} - I_{zz})\,\dot{\theta}\ \dot{\psi} - k_{fax}\,\dot{\phi}^2 - J_r\,\dot{\theta}\,\omega_{\mathrm{r}} + M_x] \\ \ddot{y} = \frac{1}{m}\ [(\mathrm{s}\,\theta\,\mathrm{s}\,\Psi\,\mathrm{c}\,\emptyset - \mathrm{c}\,\Psi\,\mathrm{s}\,\emptyset)\,F_p] & \ddot{\theta} = \frac{1}{I_{yy}}\ [(I_{zz} - I_{xx})\,\dot{\phi}\,\dot{\psi} - k_{fay}\,\dot{\theta}^2 + J_r\,\dot{\phi}\,\omega_{\mathrm{r}} + M_y] \\ \ddot{z} = -\frac{U_p}{m}\ [(\mathrm{c}\,\theta\ \mathrm{c}\,\phi)\ ] - \frac{k_{ftz}}{m}\dot{z} + g & \ddot{\psi} = \frac{1}{I_{zz}}\ [(I_{xx} - I_{yy})\,\dot{\phi}\,\dot{\theta}\ - k_{faz}\,\dot{\psi}^2 + M_z] \end{array} \right\} \tag{2.38}$$

On note que ¸

$$\begin{cases} U_x = \mathrm{s}\,\theta\,\mathrm{c}\,\Psi\,\mathrm{c}\,\emptyset + \mathrm{s}\,\Psi\,\mathrm{s}\,\emptyset \\ U_y = \mathrm{s}\,\theta\,\mathrm{s}\,\Psi\,\mathrm{c}\,\emptyset - \mathrm{c}\,\Psi\,\mathrm{s}\,\emptyset \end{cases} \tag{2.39}$$

$$\begin{bmatrix} F_p \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Cela étant, on peut négliger les coefficients aérodynamiques : k_{fax} , k_{fay}, et k_{faz} et le coefficient de frottement k_{ftz} due à leurs faibles valeurs (négligeable) :

$$\begin{cases} \ddot{x} = \frac{U_1}{m}\, U_x \\ \ddot{y} = \frac{U_1}{m}\, U_y \\ \ddot{z} = g - \frac{U_1}{m}\ [(\mathrm{c}\,\theta\ \mathrm{c}\,\phi)\ ] \end{cases} \begin{cases} \ddot{\phi} = \frac{1}{I_{xx}}\ \left[\left(I_{yy} - I_{zz}\right)\dot{\theta}\ \dot{\psi} - J_r\,\dot{\theta}\,\omega_{\mathrm{r}} + U_2\right] \\ \ddot{\theta} = \frac{1}{I_{yy}}\ \left[\left(I_{zz} - I_{xx}\right)\dot{\phi}\,\dot{\psi} + J_r\,\dot{\phi}\,\omega_{\mathrm{r}} + U_3\right] \\ \ddot{\psi} = \frac{1}{I_{zz}}\ \left[\left(I_{xx} - I_{yy}\right)\dot{\phi}\,\dot{\theta}\ +\ U_4\right] \end{cases} \tag{2.40}$$

Le système (2.43) peut être réécrit dans l'espace d'état sous la forme $\dot{X} = f\ (X, U)$,

Avec : U : vecteur d'entrée, X : vecteur d'état.

$$X = [x_1, \ldots\ldots, x_{12}]^T = \left[\phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ z\ \dot{z}\ x\ \dot{x}\ y\ \dot{y}\right]^T$$

$$\dot{X} = f(X, U) = \begin{bmatrix} x_2 \\ a_1\, x_4\, x_6 + a_3\, x_4\, \omega_{\mathrm{r}} + b_1\, U_2 \\ x_4 \\ a_4\, x_2\, x_6 + a_6\, x_2\, \omega_{\mathrm{r}} + b_2\, U_3 \\ x_6 \\ a_7\, x_2\, x_4 + b_3\, U_4 \\ x_8 \\ g + \frac{U_1}{m}\, C\, x_1\, C\, x_3 \\ x_{10} \\ \frac{U_1}{m}\, U_x \\ x_{12} \\ \frac{U_1}{m}\, U_y \end{bmatrix} \tag{2.41}$$

avec :

$$\begin{cases} a_1 = \frac{1}{I_{xx}}\left(I_{yy} - I_{zz}\right) & a_3 = \frac{-J_r}{I_{xx}} & b_1 = \frac{1}{I_{xx}} \\ a_4 = \frac{1}{I_{yy}}\left(I_{zz} - I_{xx}\right) & a_6 = \frac{J_r}{I_{yy}} & b_2 = \frac{1}{I_{yy}} \\ a_7 = \frac{1}{I_{zz}}\left(I_{xx} - I_{yy}\right) & b_3 = \frac{1}{I_{zz}} & \end{cases}$$

2.5 Conclusion

Dans ce chapitre, on a décrit les mouvements de base du quadrotor, puis on a élaboré un modèle mathématique sous forme d'équations différentielles du système en utilisant le formalisme de Newton-Euler. Par la suite, cette modélisation dynamique, sera utile pour appliquer la loi de commande la plus adéquate permettant la stabilisation du système. Le chapitre suivant porte sur la commande robuste, soit la commande backstepping adaptative qui sera bien détaillée.

CHAPITRE III COMMANDE PAR LA TECHNIQUE DU BACKSTEPPING ADAPTATIVE

3.1 Introduction

Le problème de la non-linéarité crée un défi quant au contrôle des systèmes robotiques. Au premier abord, il est nécessaire de concevoir un contrôle non linéaire auto réglable. Par ailleurs, il est indispensable de mesurer les changements de paramètres qui se produisent dans le système dynamique. Pour cet objectif, et dans ce chapitre, on va appliquer l'approche de contrôle du backstepping qui dépend de la théorie de Lyapunov et qui convient bien aux systèmes sous-actionnés La conception du contrôle backstepping offre la stabilité contrôlée pour les systèmes non linéaires. Mais, cette approche a des inconvénients majeurs tels que : la difficulté à trouver un candidat de Lyapunov approprié ainsi que les problèmes de sensibilité à des paramètres incertains. Dans ce but et pour remédier à l'incertitude des paramètres, une loi d'adaptation sera associée à l'approche backstepping pour la rendre robuste et plus efficace. Et à la fin, la performance de cette loi robuste sera illustrée par les résultats de simulations à l'aide du logiciel Matlab/Simulink.

3.2 Synthèse de la commande par backstepping

La commande par backstepping est une loi de commande pour les systèmes non linéaires basée sur le théorème de Lyapunov. Elle s'applique sur des systèmes de la forme cascade triangulaire. L'objectif de cette technique est de calculer en plusieurs étapes une loi de commande qui assure la stabilité globale du système (Madani and Benallegue 2006).

Nous proposons d'appliquer la technique de backstepping pour le contrôle du quadrotor.

❖ **Commande de rotation (Autour de l'axe x) :**

Dans cette section, on veut établir une commande U_2 qui va orienter le quadrator par l'angle ϕ et la direction de l'axe y avec U_1 qui va la propulser (voir l'équation d'état 2.40 pour la définition des variables d'état $x_1, \dots\dots, x_{12}$).

➢ Première itération

Pour avoir une sortie qui suit une trajectoire désirée ϕ_d, on définit l'erreur de trajectoire :

$$\begin{aligned} e_{x1} &= \phi_d - \phi \\ \dot{e}_{x1} &= \dot{\phi}_d - \dot{\phi} \end{aligned} \tag{3.1}$$

On utilise le théorème de Lyapunov en considérant une fonction de Lyapunov définie positive et sa dérivée par rapport au temps semi-définie négative :

$$\begin{aligned} V(e_{x1}) &= \frac{1}{2}\, {e^2}_{x1} \\ \dot{V}(e_{x1}) &= e_{x1}\, \dot{e}_{x1} = e_{x1}\, (\dot{\phi}_d - x_2) \end{aligned} \tag{3.2}$$

L'état x_2 est ensuite utilisé comme commande intermédiaire afin de satisfaire la condition de Lyapunov $\dot{V}(e_{x1}) = -\, k_{x1} {e^2}_{x1} \leq 0$ avec $k_{x1} > 0$,

On définit pour cela une commande virtuelle x_{2d} :

$$\begin{aligned} x_{2d} &= \dot{\phi}_d + k_{x1}\, e_{x1} \\ \dot{x}_{2d} &= \ddot{\phi}_d + k_{x1}\, \dot{e}_{x1} \end{aligned} \tag{3.3}$$

➢ Deuxième itération

À cette étape une nouvelle erreur engendre :

$$e_{x2} = x_{2d} - x_2 = \dot{\phi}_d + k_{x1}\, e_{x1} - \dot{\phi} \qquad (3.4)$$
$$\dot{e}_{x2} = \ddot{\phi}_d - \ddot{\phi} + k_{x1}\, \dot{e}_{x1}$$

À partir des équations (3.2) et (3.4), on obtient :

$$\dot{e}_{x1} = e_{x2} - k_{x1}\, e_{x1} \qquad (3.5)$$

Pour éliminer cette erreur, la fonction candidate de Lyapunov $V(e_{x1})$ est augmentée d'un autre terme, qui va prendre en charge la nouvelle erreur qui a été introduite précédemment :

$$V(e_{x1}, e_{x2}) = \frac{1}{2}\,(e^2{}_{x1} + e^2{}_{x2}) \qquad (3.6)$$
$$\dot{V}(e_{x1}, e_{x2}) = e_{x1}\, \dot{e}_{x1} + e_{x2}\, \dot{e}_{x2}$$

En remplaçant les expressions (3.4) et (3.5) dans l'équation (3.6), on obtient :

$$\dot{V}(e_{x1}, e_{x2}) = e_{x1}\ (e_{x2} - k_{x1}\, e_{x1}\ ) + e_{x2}\,(\ddot{\phi}_d - \ddot{\phi} + k_{x1}\, \dot{e}_{x1}) \qquad (3.7)$$
$$\dot{V}(e_{x1}, e_{x2}) = -\ k_{x1}\, e^2{}_{x1} + e_{x2}\,[\ddot{\phi}_d - \ \ddot{\phi} + k_{x1}\,(e_{x2} - k_{x1}\, e_{x1}\,) + e_{x1}]$$

En remplaçant l'expression de $\ddot{\phi}$ dans l'équation (3.7), on obtient :

$$\dot{V}(e_{x1}, e_{x2}) = -\ k_{x1}\, e^2{}_{x1} + e_{x2}\,[\ddot{\phi}_d - a_1 x_4 x_6 - b_1 U_2 + k_{x1}\,(e_{x2} \qquad (3.8)$$
$$- k_{x1}\, e_{x1}\,) + e_{x1}]$$

Afin de satisfaire la condition de Lyapunov ($\dot{V}(e_{x1}, e_{x2}) \leq 0$) , on pose l'expression suivante :

$$\ddot{\phi}_d - a_1 x_4 x_6 - b_1 U_2 + k_{x1}\,(e_{x2} - k_{x1}\,e_{x1}\,) + e_{x1} = -k_{x2}\,e_{x2} \qquad (3.9)$$

$$U_2 = \frac{1}{b_1}[-a_1 x_4 x_6 - a_3\,x_4 + \ddot{\phi}_d + k_{x1}\,(-k_{x1} e_{x1} + e_{x2}) + k_{x2}\,e_{x2} + e_{x1}]$$

de sorte que $\left(\dot{V}(e_{x1}, e_{x2}\right) = -\ k_{x1}\,{e^2}_{x1} - \ k_{x2}\,{e^2}_{x2} \leq 0)$, ce qui assure la stabilité de l'orientation du quadrator de l'angle ϕ.

❖ **Commande de rotation (Autour de l'axe y) :**

On suit les mêmes étapes pour déterminer la commande U_3:

$$U_3 = \frac{1}{b_2}[-a_4 x_2 x_6 - a_6\,x_2 + \ddot{\theta}_d + k_{x3}\,(-k_{x3} e_{x3} + e_{x4}) + k_{x4} e_{x4} + e_{x3}] \qquad (3.10)$$

❖ **Commande de rotation (Autour de l'axe z) :**

On suit les mêmes étapes pour déterminer la commande U_4 :

$$U_4 = \frac{1}{b_3}[-a_7 x_2 x_4 + \ddot{\psi}_d + k_{x5}(-\,k_{x5}\,e_{x5} + e_{x6}) + k_{x6}\,e_{x6} + e_{x5}] \qquad (3.11)$$

❖ **Commande pour mouvement suivant l'axe x :**

On veut commander le mouvement x en utilisant la technique de backstepping. Le modèle est donné par le modèle d'état (2.41).

➢ Première itération

Pour avoir une sortie qui suit une trajectoire désirée x_d, on définit l'erreur de trajectoire :

$$\begin{aligned} e_{x9} &= x_d - x \\ \dot{e}_{x9} &= \dot{x}_d - \dot{x} \end{aligned} \qquad (3.12)$$

On utilise le théorème de Lyapunov en considérant une fonction de Lyapunov définie positive et sa dérivée par rapport au temps semi-définie négative :

$$V(e_{x9}) = \frac{1}{2}\, e^2{}_{x9} \qquad (3.13)$$
$$\dot{V}(e_{x9}) = e_{x9}\, \dot{e}_{x9} = e_{x9}\, (\dot{x}_d - x_{10})$$

L'état x_{10} est ensuite utilisé comme une commande intermédiaire afin de satisfaire la condition de Lyapunov $\dot{V}(e_{x9}) = -\, k_{x9} e^2{}_{x9} \leq 0$, avec $k_{x9} > 0$,

On définit pour cela une commande virtuelle x_{10d}, présenté par l'expression suivante :

$$x_{10d} = \dot{x}_d + k_{x9}\, e_{x9} \qquad (3.14)$$
$$\dot{x}_{10d} = \ddot{x}_d + k_{x9}\, \dot{e}_{x9}$$

- Deuxième itération

À cette étape une nouvelle erreur engendre :

$$e_{x10} = x_{10d} - x_{10} = \dot{x}_d + k_{x9}\, e_{x9} - \dot{x} \qquad (3.15)$$
$$\dot{e}_{x10} = \ddot{x}_d - \ddot{x} + k_{x9}\, \dot{e}_{x9}$$

À partir des équations (3.12) et (3.15), on obtient :

$$\dot{e}_{x9} = e_{x10} - k_{x9}\, e_{x9} \qquad (3.16)$$

Pour éliminer cette erreur, une fonction candidate de Lyapunov $V(e_{x9})$ est augmentée d'un autre terme, qui va prendre en charge la nouvelle erreur qui a été introduite précédemment :

$$V(e_{x9}, e_{x10}) = \frac{1}{2}\,(e^2{}_{x9} + e^2{}_{x10}) \qquad (3.17)$$
$$\dot{V}(e_{x9}, e_{x10}) = e_{x9}\,\dot{e}_{x9} + e_{x10}\,\dot{e}_{x10}$$

En remplaçant les expressions (3.15) et (3.16) dans l'équation (3.14), on obtient :

$$\dot{V}(e_{x9}, e_{x10}) = e_{x9}\ (-k_{x9}\,e_{x9}+e_{x10})\ + e_{x10}\,(\ddot{x}_d - \ddot{x} + k_{x9}\,\dot{e}_{x9}) \qquad (3.18)$$
$$\dot{V}(e_{x9}, e_{x10}) = -\ k_{x9}\,e^2{}_{x9}\ + e_{x10}\,[\ddot{x}_d - \ddot{x} + k_{x9}\,(e_{x10} - k_{x9}\,e_{x9}\,) + e_{x9}]$$

En remplaçant l'expression de $\ddot{x}$ dans l'équation (3.18), on obtient :

$$\dot{V}(e_{x9}, e_{x10}) = -\ k_{x9}\,e^2{}_{x9} + e_{x10}\,[\ddot{x}_d - \frac{U_1}{m}U_X + k_{x9}\,(e_{x10} - k_{x9}\,e_{x9}\,) + e_{x9}\,] \qquad (3.19)$$

Afin de satisfaire la condition de Lyapunov ($\dot{V}(e_{x9}, e_{x10}) \leq 0$) , on pose l'expression suivante :

$$\ddot{x}_d - \frac{U_1}{m}U_X + k_{x9}\,(e_{x10} - k_{x9}\,e_{x9}\,) + e_{x9} = -k_{x10}\,e_{x10} \qquad (3.20)$$
$$U_x = \frac{m}{U_1}\,[\ddot{x}_d + +k_{x9}\,(e_{x10} - k_{x9}\,e_{x9}\,) + k_{x10}\,e_{x10} + e_{x9}]$$

de sorte que $\left(\dot{V}(e_{x9}, e_{x10}\right) = -\ k_{x9}\,e^2{}_{x9} - \ k_{x10}\,e^2{}_{x10} \leq 0)$, ce qui assure la stabilité de l'axe x.

❖ **Commande pour mouvement suivant l'axe y :**

On suit les mêmes étapes pour déterminer la commande U_y :

$$U_y = \frac{m}{U_1}[\ddot{y}_d + e_{x11}(1 - k^2{}_{x11}) - \ e_{x12}\,(k_{x11} + k_{x12})] \qquad (3.21)$$

❖ **Commande pour mouvement suivant l'axe z :**

On suit les mêmes étapes pour déterminer la commande U_1 du quadrator :

➢ Première itération

Pour avoir une sortie qui suit une trajectoire désirée z_d, on définit l'erreur de trajectoire :

$$\begin{aligned} e_{x7} &= z_d - \mathrm{z} \\ \dot{e}_{x7} &= \dot{z}_d - \dot{z} \end{aligned} \tag{3.22}$$

On utilise le théorème de Lyapunov en considérant une fonction de Lyapunov définie positive et sa dérivée par rapport au temps semi-définie négative :

$$\begin{aligned} V(e_{x7}) &= \frac{1}{2}\, {e^2}_{x7} \\ \dot{V}(e_{x7}) &= e_{x7}\, \dot{e}_{x7} = e_{x7}\, (\dot{z}_d - \dot{z}) \end{aligned} \tag{3.23}$$

L'état x_8 est utilisé comme une commande intermédiaire afin de satisfaire la condition de Lyapunov $\dot{V}(e_{x7}) = -\, k_{x7} {e^2}_{x7} \leq 0$, avec $k_{x7} > 0$,

On définit pour cela une commande virtuelle x_{8d} :

$$\begin{aligned} x_{8d} &= \dot{z}_d + k_{x7}\, e_{x7} \\ \dot{x}_{8d} &= \ddot{x}_d + k_{x9}\, \dot{e}_{x9} \end{aligned} \tag{3.24}$$

➢ Deuxième itération

À cette étape une nouvelle erreur engendre :

$$\begin{aligned} e_{x8} &= \dot{z}_d - \dot{z} + k_{x7}\, e_{x7} \\ \dot{e}_{x8} &= \ddot{z}_d - \ddot{z} + k_{x7}\, \dot{e}_{x7} \end{aligned} \tag{3.25}$$

À partir des équations (3.22) et (3.25), on obtient :

$$\dot{e}_{x7} = -k_{x7}\, e_{x7} + e_{x8} \tag{3.26}$$

Pour éliminer cette erreur, une fonction candidate de Lyapunov $V(e_{x7})$ est augmentée d'un autre terme, qui va prendre en charge la nouvelle erreur qui a été introduite précédemment :

$$\begin{aligned} V(e_{x7}, e_{x8}) &= \frac{1}{2}\,({e^2}_{x7} + {e^2}_{x8}) \\ \dot{V}(e_{x7}, e_{x8}) &= e_{x7}\,\dot{e}_{x7} + e_{x8}\,\dot{e}_{x8} \end{aligned} \tag{3.27}$$

En remplaçant les expressions (3.25) et (3.26) dans l'équation (3.27), on obtient :

$$\begin{aligned} \dot{V}(e_{x7}, e_8) &= e_{x7}\ (-k_{x7}\, e_{x7} + e_{x8})\ + e_{x8}\,(\ddot{z}_d - \ddot{z} + k_{x7}\,\dot{e}_{x7}) \\ \dot{V}(e_{x7}, e_{x8}) &= -\ k_{x7}\, {e^2}_{x7} + e_{x8}\,[\ddot{z}_d - \ddot{z} + k_{x7}\,(-k_{x7}\, e_{x7} + e_{x8}\,) + e_{x7}] \end{aligned} \tag{3.28}$$

En remplaçant l'expression de $\ddot{z}$ dans l'équation (3.28), on obtient :

$$\begin{aligned} \dot{V}(e_{x7}, e_{x8}) = -\ k_{x7}\, {e^2}_{x7} + e_{x8}\,[\ddot{z}_d - g + \frac{U_1}{m}\, cx_1 cx_3 \\ + k_{x7}\,(-k_{x7}\, e_{x7} + e_{x8}\,) + e_{x7}] \end{aligned} \tag{3.29}$$

$$\begin{aligned} \dot{V}(e_{x7}, e_{x8}) = -\ k_{x7}\, {e^2}_{x7} +\ e_{x8}\,[\ddot{z}_d - g + \frac{U_1}{m} cx_1 cx_3 \\ + k_{x7}\,(-k_{x7}\, e_{x7} + e_{x8}\,) + e_{x7}] \end{aligned} \tag{3.30}$$

Afin de satisfaire la condition de Lyapunov ($\dot{V}(e_{x7}, e_{x8}) \leq 0$) , on pose l'expression suivante :

$$\ddot{z}_d - g + \frac{U_1}{m}\, cx_1 cx_3 + k_{x7}\,(-k_{x7}\, e_{x7} + e_{x8}\,) + e_{x7} = -k_{x8}\, e_{x8} \tag{3.31}$$
$$U_1 = \frac{m}{cx_1 cx_3}[g - \ddot{z}_d - k_{x7}\,(-k_{x7} e_{x7} + e_{x8}\,) - k_{x8}\, e_{x8} - e_{x7}]$$

de sorte que $\left(\dot{V}(e_{x7}, e_{x8}\right) = -\ k_{x7}\, e^2{}_{x7} -\ k_{x8}\, e^2{}_{x8} \leq 0)$, ce qui assure la stabilité de l'axe z.

❖ **Commande des angles désirée :**

En se basant sur (2.39), les commandes axillaires U_x et U_y, sont utilisées pour calculer les angles souhaités :$\emptyset_d$ et θ_d. Les angles désirées $\emptyset_d$ et $\theta_d$sont représentées par (3.32) et (3.33).

$$\emptyset_d = arcsin[U_x sin\psi_d - U_y cos\psi_d] \tag{3.32}$$

$$\theta_d = arcsin[\frac{U_x\, sin\psi_d + U_y\, sin\psi_d}{cos\phi_d}] \tag{3.33}$$

3.3 Synthèse de la commande Backstepping Adaptative :

Dans cette section, on propose une commande par backstepping adaptative dont le but est de minimiser l'erreur de suivi et d'assurer la stabilité de Lyapunov. Dans cette commande adaptative, la commande backstepping sera appliquée sur le système quadrotor en supposant que les paramètres sont inconnus afin d'assurer la robustesse et atténuer les effets des perturbations externes (Benaskeur 2000), (Boudguiga 2016).

La commande Backstepping adaptative est une technique où les paramètres de commande varient en temps réel pour s'adapter aux changements dans la dynamique du système. Ces changements peuvent être dus à des erreurs de modélisation, à des incertitudes paramétriques ou à des perturbations externes.

❖ **Commande de rotation (Autour de l'axe x) :**

Soient $\hat{a}_1$ et $\hat{b}_1$ les estimations de a_1 et b_1, respectivement. Alors la dynamique (2.41) devient :

$$\left\langle \begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = a_1\, x_2\, x_6 + b_1\, U_2 \end{cases} \middle| \begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = (\tilde{a}_1 + \hat{a}_1)\, x_2\, x_6 + (\tilde{b}_1 + \hat{b}_1)\, U_2 \end{cases} \right\rangle \tag{3.34}$$

Avec : $\tilde{a}_1 = a_1 - \hat{a}_1$ et $\tilde{b}_1 = b_1 - \hat{b}_1$ sont les erreurs d'estimations.

➢ Première itération

Pour avoir une sortie qui une trajectoire désirée ϕ_d, on définit l'erreur de trajectoire :

$$\left\langle \begin{matrix} e_{x1} = x_1 - x_{1d} = \phi_d - \phi \\ \dot{e}_{x1} = x_2 - \dot{x}_{1d} = \dot{\phi}_d - \dot{\phi} \\ V(e_{x1}) = \frac{1}{2}\, {e^2}_{x1} \end{matrix} \middle| \dot{V}(e_{x1}) = e_{x1}\, \dot{e}_{x1} = e_{x1}\, (\dot{x}_{1d} - x_2) \right\rangle \tag{3.35}$$

L'état x_2 est ensuite utilisé comme une commande intermédiaire pour satisfaire la condition de Lyapunov $\dot{V}(e_{x1}) = -\, k_{x1} {e^2}_{x1} \leq 0$, avec $k_{x1} > 0$, on définit la commande virtuelle x_{2d}:

$$\begin{matrix} x_{2d} = \dot{\phi}_d - k_{x1}\, e_{x1} \\ \dot{x}_{2d} = \ddot{\phi}_d - k_{x1}\, \dot{e}_{x1} \end{matrix} \tag{3.36}$$

➢ Deuxième itération

À cette étape une nouvelle erreur engendre :

$$\left\langle \begin{matrix} e_{x2} = x_2 - x_{2d} = \dot{\phi}_d - \dot{\phi} + k_{x1}\, e_{x1} \\ \dot{e}_{x2} = \ddot{\phi}_d - \ddot{\phi} + k_{x1}\, \dot{e}_{x1} \end{matrix} \right. \tag{3.37}$$

À partir des équations (4.35) et (4.37), on obtient :

$$\dot{e}_{x1} = e_{x2} - k_{x1}\, e_{x1} \tag{3.38}$$

À partir des équations (4.34) et (4.37) et (3.38) on obtient :

$$\dot{e}_{x2} = \ddot{\phi}_d - \ddot{\phi} + k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1}) \tag{3.39}$$

$$\dot{e}_{x2} = \ddot{\phi}_d - (\tilde{a}_1 + \hat{a}_1)\, x_2\, x_6 - (\tilde{a}_3 + \hat{a}_3)\;\omega_{\mathrm{r}}\, x_4 - (\tilde{b}_1 + \hat{b}_1)\;U_2 - + k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1}) \tag{3.40}$$

Pour faire la loi d'adaptation de $a_1$et calculer U_2, on fait un changement de variable.

Soit $b_1 U_2 = \overline{U}_2 - b_1\, \tilde{b}_1\, U_2$

$$\dot{e}_{x2} = \ddot{\phi} - \ddot{\phi}_d + k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1})) \tag{3.41}$$

$$\dot{e}_{x2} = (\tilde{a}_1 + \hat{a}_1)\, x_4\, x_6 + \;(\tilde{b}_1 + \hat{b}_1)\;U_2 - \ddot{\phi}_d + k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1})$$

$$\dot{e}_{x2} = (\tilde{a}_1 + \hat{a}_1)\, x_4\, x_6 + \;\overline{U}_2 - b_1\, \tilde{b}_1\, U_2 - \ddot{\phi}_d + k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1})$$

On choisit $\overline{U}_2$ de la forme suivante :

$$\overline{U}_2 = V_2\; - \hat{a}_1\, x_4\, x_6 + \ddot{\phi}_d - k_{x1}\,(\, e_{x2} - k_{x1}\, e_{x1}) \tag{3.42}$$

On remplace $\overline{U}_2$ dans (4.41), on obtient la forme suivante :

$$\dot{e}_{x2} = V_2 - b_1\, \tilde{b}_1\, \overline{U}_2 + \tilde{a}_1\, x_4\, x_6 \tag{3.43}$$

On définit V_2 sous la forme suivante :

$$V_2 = \frac{1}{2}e^2{}_{x1} + \frac{1}{2}e^2{}_{x2} + \frac{1}{2\gamma}\tilde{a}^2{}_1 + \frac{b_1}{2\gamma}\tilde{b}^2{}_1 \tag{3.44}$$

$$\dot{V}_2 = e_{x1}\,\dot{e}_{x1} + e_{x2}\,\dot{e}_{x2} + \frac{1}{\gamma}\,\tilde{a}_1\,\dot{\tilde{a}}_1 + \frac{b_1}{\gamma}\,\tilde{b}_1\,\dot{\tilde{b}}_1$$

$$\dot{V}_2 = e_{x1}\,(e_{x2} - k_{x1}\,e_{x1}) + e_{x2}\,(V_2 - b_1\,\tilde{b}_1\,\bar{U}_2 + \tilde{a}_1\,x_4\,x_6) + \frac{1}{\gamma}\,\tilde{a}_1\,\dot{\tilde{a}}_1 + \frac{b_1}{\gamma}\,\tilde{b}_1\,\dot{\tilde{b}}_1$$

$$\dot{V}_2 = e_{x1}\,(e_{x2} - k_{x1}\,e_{x1}) + e_{x2}\,V_2 + \tilde{b}_1\left(-b_1\,\bar{U}_2\,e_{x2} + \frac{b_1}{\gamma}\,\dot{\tilde{b}}_1\right) + \tilde{a}_1\left(\,e_{x2}\,x_4\,x_6 + \frac{1}{\gamma}\,\dot{\tilde{a}}_1\right)$$

$$\dot{V}_2 = -k_{x1}e^2{}_{x1} + e_{x2}\,(V_2 + e_{x1}) + \tilde{b}_1\left(-b_1\,\bar{U}_2\,e_{x2} + \frac{b_1}{\gamma}\,\dot{\tilde{b}}_1\right) + \tilde{a}_1\left(\,e_{x2}\,x_4\,x_6 + \frac{1}{\gamma}\,\dot{\tilde{a}}_1\right)$$

Pour satisfaire la condition de Lyapunov $\dot{V}_2 \leq 0$, avec $k_{x1} > 0$, on obtient les conditions suivantes :

$$\left\langle\left\langle \begin{array}{l} e_{x2}\,(V_2 + e_{x1}) = -k_{x2}e^2{}_{x2} \\ \tilde{b}_1\left(-b_1\,\bar{U}_2\,e_{x2} + \frac{b_1}{\gamma}\,\dot{\tilde{b}}_1\right) = 0 \\ \tilde{a}_1\left(\,e_{x2}\,x_4\,x_6 + \frac{1}{\gamma}\,\dot{\tilde{a}}_1\right) = 0 \end{array} \middle| \begin{array}{c} V_2 = -e_{x1} - k_{x2}\,e_{x2}) \\ \dot{\tilde{b}}_1 = \gamma\,e_{x2}\,\overline{U_2} \\ \dot{\tilde{a}}_1 = -\gamma\;\,x_4 x_6\,e_{x2} \end{array} \right\rangle\right\rangle \tag{3.45}$$

On remplace l'expression de V_2 représenté par (4.45) dans (4.42), on obtient l'expression de la commande suivante :

$$\bar{U}_2 = -e_{x1} - k_{x2}\,e_{x2} - \hat{a}_1\,x_4\,x_6 + \ddot{\phi}_d - k_{x1}\,(\,e_{x2} - k_{x1}\,e_{x1}) \tag{3.46}$$

❖ **Commande de rotation (Autour de l'axe y) :**

On suit les mêmes étapes pour déterminer la commande $\bar{U}_3$:

$$\bar{U}_3 = -e_{x3} - k_{x4}\,e_{x4} - \hat{a}_3\,x_2\,x_6 + \ddot{\theta}_d - k_{x3}\,(\,e_{x4} - k_{x3}\,e_{x3}) \tag{3.47}$$

$$\begin{cases} \dot{\tilde{b}}_2 = \gamma\,e_{x4}\,\overline{U_3} \\ \dot{\tilde{a}}_3 = -\gamma\;\,x_2 x_6\,e_{x4} \end{cases}$$

❖ **Commande de rotation (Autour de l'axe z) :**

On suit les mêmes étapes pour déterminer la commande $\overline{U_4}$:

$$\overline{U_4} = -e_{x5} - k_{x6}\, e_{x6} \; - \hat{a}_5\, x_2\, x_4 + \ddot{\psi}_d - k_{x5}\,(\, e_{x6} - k_{x5}\, e_{x5}) \tag{3.48}$$
$$\begin{cases} \dot{\tilde{b}_3} = \gamma\, e_{x6}\, \overline{U_4} \\ \dot{\tilde{a}_5} = -\gamma\;\; x_2 x_4\, e_{x6} \end{cases}$$

❖ **Commande pour mouvement suivant l'axe z :**

On suit les mêmes étapes pour déterminer la commande U_1 :

$$\overline{U_1} = -e_{x7} - k_{x8}\, e_{x8} \; - k_{x7}\,(\, e_{x8} - k_{x7}\, e_{x7}) + \ddot{z}_d + g \tag{3.49}$$
$$\begin{cases} \dot{\tilde{b}_z} = \gamma\, e_{x8}\, \overline{U_1} \\ avec\; b_z = a_6\, Cx_1 Cx_3 \end{cases}$$

❖ **Commande pour mouvement suivant l'axe x :**

On suit les mêmes étapes pour déterminer la commande U_x :

$$\overline{U_x} = -e_{x9} - k_{x10}\, e_{x10} \; - k_{x9}\,(\, e_{x10} - k_{x9}\, e_{x9}) + \ddot{x}_d \tag{3.50}$$
$$\begin{cases} \dot{\tilde{b}_x} = \gamma\, e_{x10}\, \overline{U_x} \\ avec\; b_x = a_6\, U_1 \end{cases}$$

❖ **Commande pour mouvement suivant l'axe y :**

On suit les mêmes étapes pour déterminer la commande U_y :

$$\overline{U_y} = -e_{x11} - k_{x12}\, e_{x12} \; - k_{x11}\,(\, e_{x12} - k_{x11}\, e_{x11}) + \ddot{y}_d \tag{3.51}$$
$$\begin{cases} \dot{\tilde{b}_y} = \gamma\, e_{x12}\, \overline{U_y} \\ avec\; b_y = a_6\, U_1 \end{cases}$$

3.4 Conclusion :

L'objectif principal de ce chapitre est la synthèse d'une loi de commande stabilisante robuste pour assurer la stabilité du système. La difficulté de son contrôle est due principalement à sa dynamique complexe, non linéaire, multi variable et surtout à son sous actionnement. Nous avons choisi la commande par backstepping adaptative. La commande proposée est capable de contrôler le système avec un bon suivi de la trajectoire et pourrait être appliquée à un modèle réel. Une validation par Matlab /Simulink de cette approche sera présentée dans le quatrième chapitre et enfin les performances seront illustrées par une étude expérimentale bien détaillée dans le cinquième chapitre.

CHAPITRE IV SIMULATIONS, RÉSULTATS ET INTERPRÉTATION

4.1 Introduction

Dans cette section, on va présenter les résultats de simulations par Backstepping et Backstepping-adaptative. Ces simulations ont été réalisées avec le logiciel Matlab / Simulink R2020b. En premier lieu, on va présenter les paramètres du PARROT MAMBO ainsi que les différents gains des contrôleurs qui seront utilisés au cours de ces simulations. Ensuite viendra les simulations, résultats et leurs interprétations. Une trajectoire désirée d'un cercle sera simulée afin de mettre en évidence l'efficacité de l'approche proposée.

4.2 Présentation de l'environnement du travail

❖ **Caractéristique du quadrotor :**

Figure 4. 1 Parrot Mambo

Le tableau suivant présente les différents paramètres du Parrot Mambo qui sont utilisés dans les simulations :

Tableau 4. 1 Paramètres du système « Quadrotor ».

Symbole	Description	Valeur
m	Masse du quadrotor	0.63 Kg
g	L'accélération gravitationnelle	9.81 m s
l	Distance entre le centre du quadrotor et axe de rotation du rotor.	0.23 m
d	Coefficient aérodynamique de trainée	$7.5\ 10^{-7}$
b	Coefficient aérodynamique de poussée	$3.13\ 10^{-5}$
J_r	Moment d'inertie du rotor	$1.0209\ 10^{-7}$
I_{xx}	Moment d'inertie autour de l'axe X	$0.6860\ 10^{-3}$
I_{yy}	Moment d'inertie autour de l'axe Y	$0.0920\ 10^{-3}$
I_{zz}	Moment d'inertie autour de l'axe Z	$0.1366\ 10^{-3}$

❖ **Environnement du travail et stratégie de contrôle :**

Le quadrotor est un système sous actionné et les quatre moteurs sont utilisés pour contrôler les six degrés de liberté x, y, z, roll, pitch et yaw. La structure de contrôle est basée sur deux boucles (Oloo and Kamau 2018), (Bouabdallah and Siegwart 2005), (Bouabdallah 2007), comme il est exposé sous le schéma synoptique suivant dans la Figure 4.2 :

Boucle interne : qui contrôle les angles roll, pitch, yaw et l'altitude z. Ce dernier utilise la valeur de référence pour générer le signal de commande approprié. Ce signal est composé de quatre commandes ($U_1, U_2, U_3\ et\ U_4$) qui sont des entrées du modèle du quadrotor.

Boucle externe : qui contrôle les positions x et y. Le but est de calculer les angles roll et pitch souhaitées sur la base de la position désirée.

Comme on ne peut pas commander x et y qu'à travers le roll (ϕ) et le pitch (θ), U_x et U_y sont considérés comme des commandes virtuelles pour le système. Ensuite, en utilisant le bloc de correction, ϕ_d et θ_d seront générées à partir de U_x et U_y pour permettre au système d'atteindre x_d et y_d.

Les entrées de commande du système du quadrotor sont : $U_1, U_2, U_3 et\ U_4$.

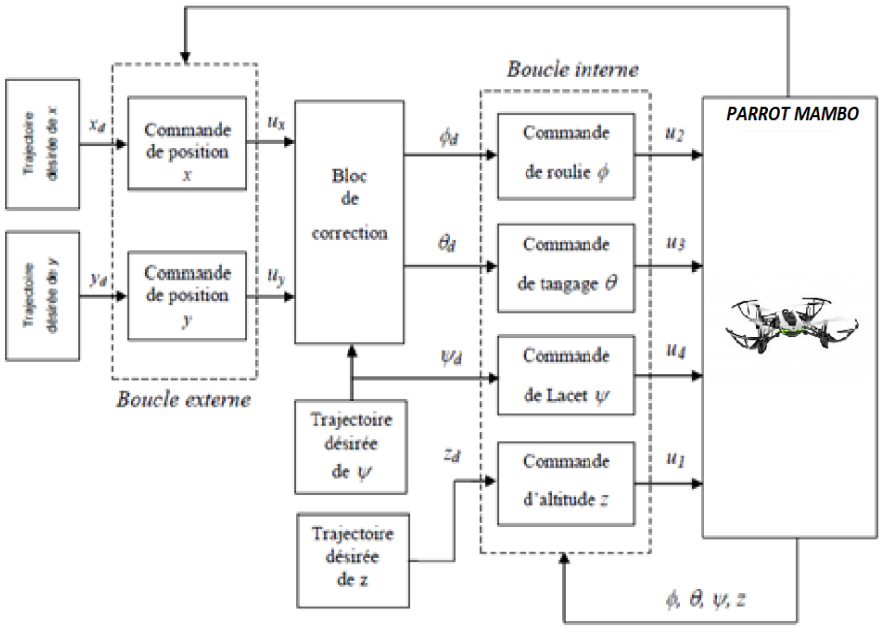

Figure 4. 2 Structure de contrôle

❖ **Trajectoire :**

La trajectoire de cercle est décrite par les équations suivantes :

$$x_d = \begin{cases} 0 & si\ t < 3seconds \\ 0{,}5 \sin(0{,}5t) & si\ t > 3seconds \end{cases}$$

$$y_d = \begin{cases} 0 & si\ t < 3seconds \\ 0{,}5 + \sin(0{,}5t) & si\ t > 3seconds \end{cases}$$

$$z_d = -1 \quad \psi_d = 0$$

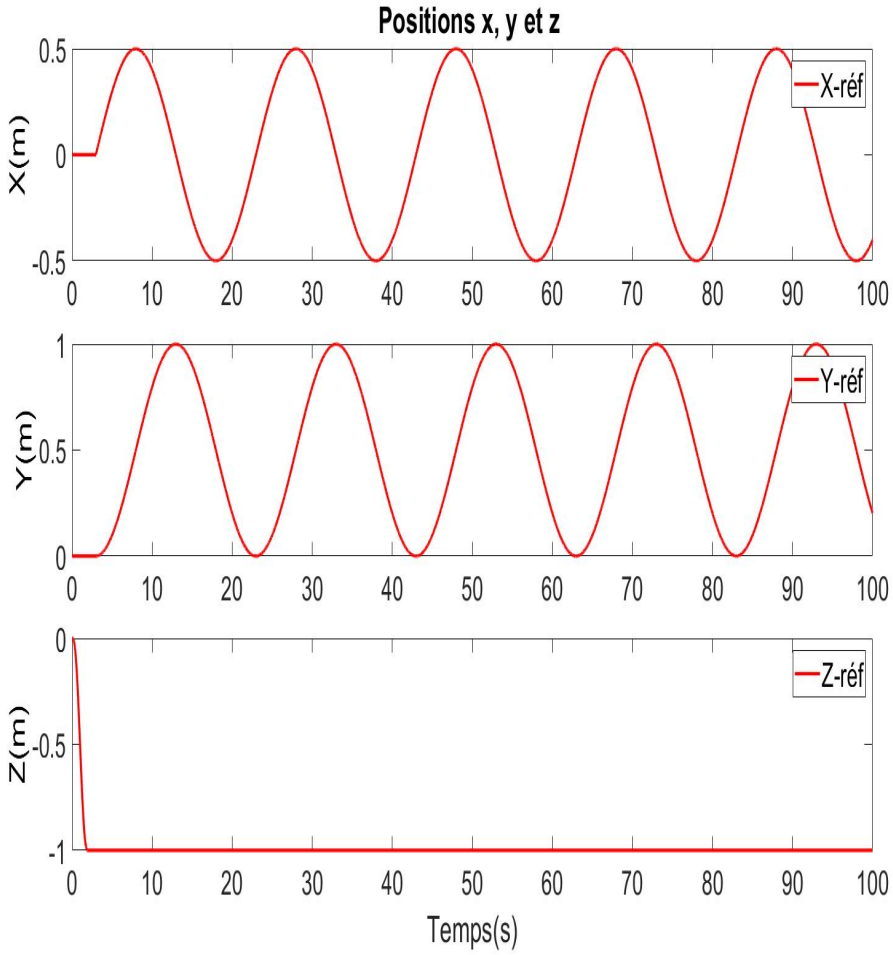

Figure 4. 3 Trajectoire d'un cercle

Les valeurs initiales de positions sont x (0) =y (0) =z (0) =0 et les valeurs initiales des angles d'Euler sont $\emptyset(0) = \theta\ (0) = \psi\ (0) = 0$.

4.3 Simulation de la commande Backstepping :

Dans cette section on va simuler le modèle avec la trajectoire du cercle pour valider la performance de notre approche de commande Backstepping à suivre les trajectoires complexes non stationnaires.

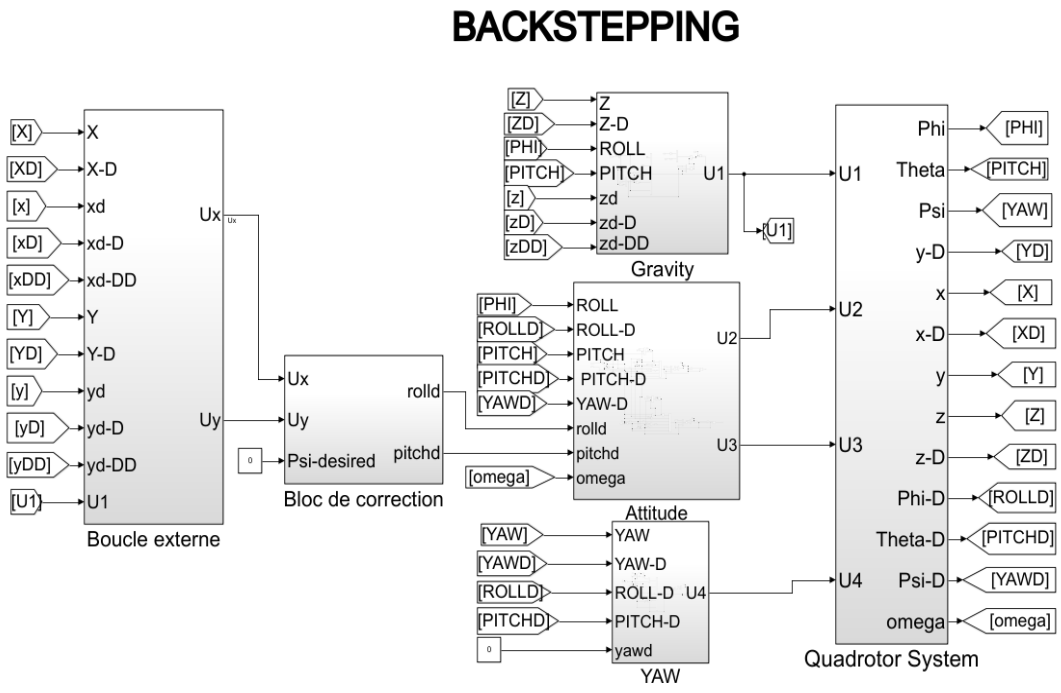

Figure 4. 4 Modèle Simulink pour Backstepping

La Figure 4.4 présente le modèle Simulink de la commande backstepping. La stratégie de commande basée principalement sur deux boules (boucle interne et boucle externe). La boucle interne contient quatre lois de commande : commande de rotation selon l'axe x par U_2 (3.9), commande de rotation selon l'axe y par U_3 (3.10), commande de rotation selon l'axe z par U_4 (3.11) et la commande de mouvement suivant l'axe z par U_1(3.31). La boucle externe inclut deux lois de commande : commande de mouvement suivant l'axe x par U_{x} (3.20) et commande de mouvement suivant l'axe y par U_y (3.21). Passant par le bloc de correction, qui contient deux commandes : la commande pour l'angle désirée $\emptyset$ par $\emptyset_d$ (3.32) et la commande pour l'angle désirée θ par θ_d (3.33).

Les entrées du modèle du quadrotor sont : $U_1, U_2, U_3\ et\ U_4$ et les sorties sont les trois position x, y et z et les trois angles de rotations $\emptyset$, θ et ψ.

Le tableau suivant illustre les gains de la commande backstepping, plusieurs essais-erreurs ont été faits pour avoir sur les bonnes valeurs :

Tableau 4. 2 Gains du Backstepping

Désignation	k_{x1}	k_{x2}	k_{x3}	k_{x4}	k_{x5}	k_{x6}	k_{x7}	k_{x8}	k_{x9}	k_{x10}	k_{x11}	k_{x12}
Valeurs	55	45	52	47	2	1.9	2.7	1.4	2.8	1.3	2.5	2.5

❖ **Résultats de simulation :**

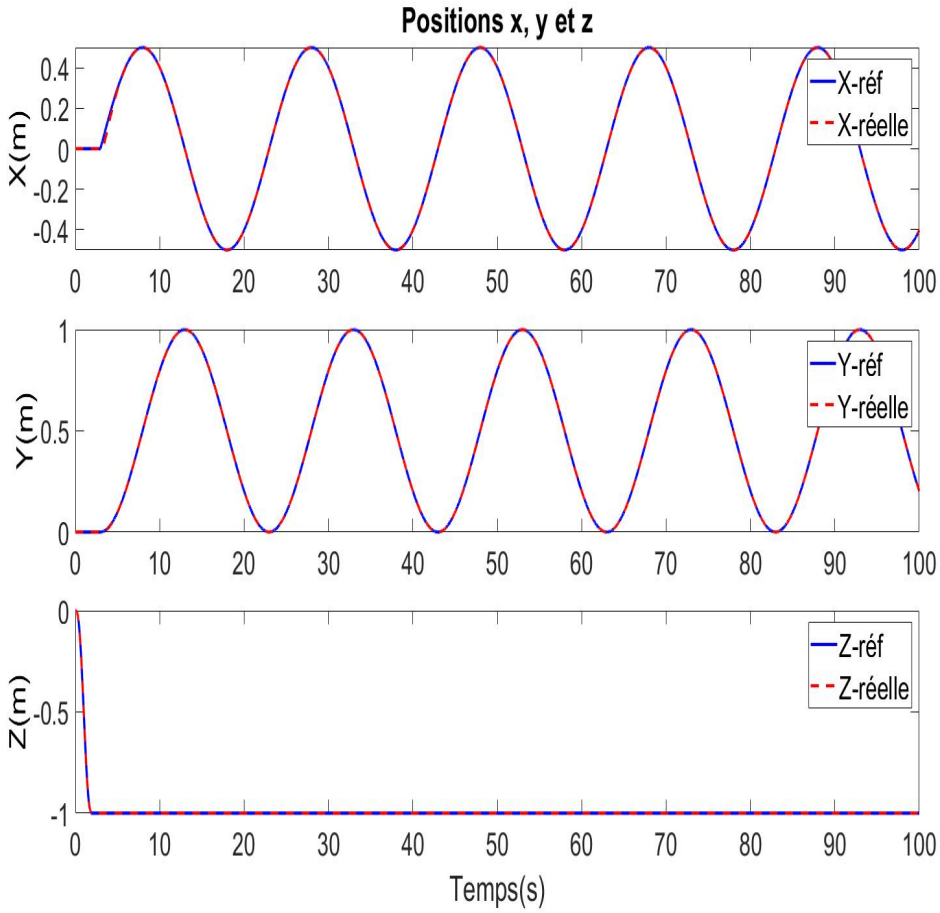

Figure 4. 5 Suivie de trajectoire du cercle pour les positions x, y et z

Les résultats obtenus comme indiqué dans la Figure 4.5, montrent clairement que le quadrotor suit la trajectoire de référence désirée, mais avec une petite erreur selon l'axe x pendant les 7 premiers secondes. La Figure 4.6 présente la réalisation d'une trajectoire cercle en 3D par la commande backstepping.

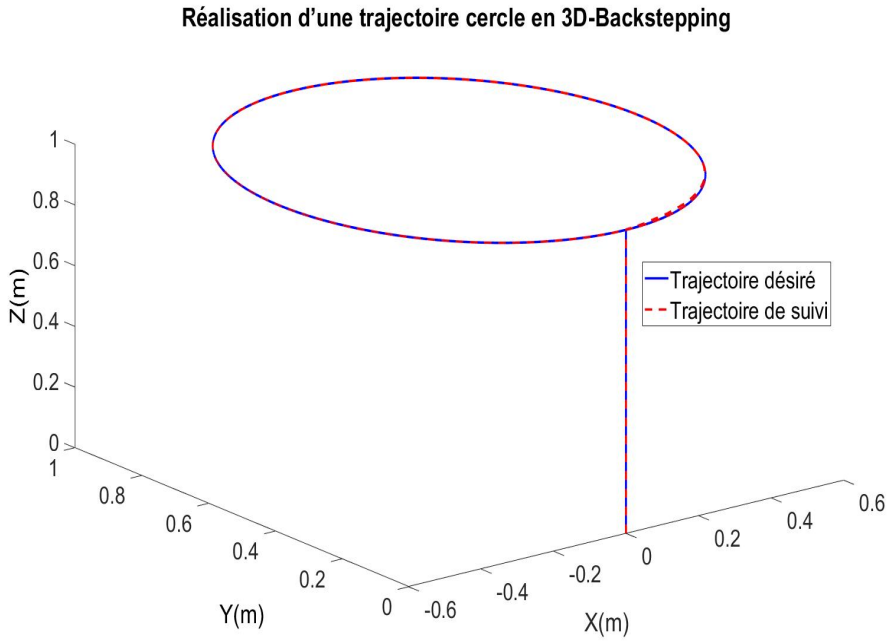

Figure 4. 6 Cercle en 3D-Backstepping

4.4 Simulation de la commande Backstepping-Adaptative

Les résultats de simulation obtenus par la commande Backstepping ont montré le suivi parfait de la trajectoire désirée, mais avec une petite erreur selon l'axe x. Afin de rendre la commande robuste, une loi d'adaptation est associée à cette commande qui estime les paramètres inconnus et qui assure leur convergence vers leurs valeurs respectives, sans affecter le bon fonctionnement du système, notamment la stabilité. À l'aide d'une deuxième simulation par Backstepping-adaptative appliqué sur le modèle Simulink comme il est montré par le modèle Simulink suivant, nous avons démontré l'efficacité de notre approche de commande à suivre une trajectoire variante dans le temps.

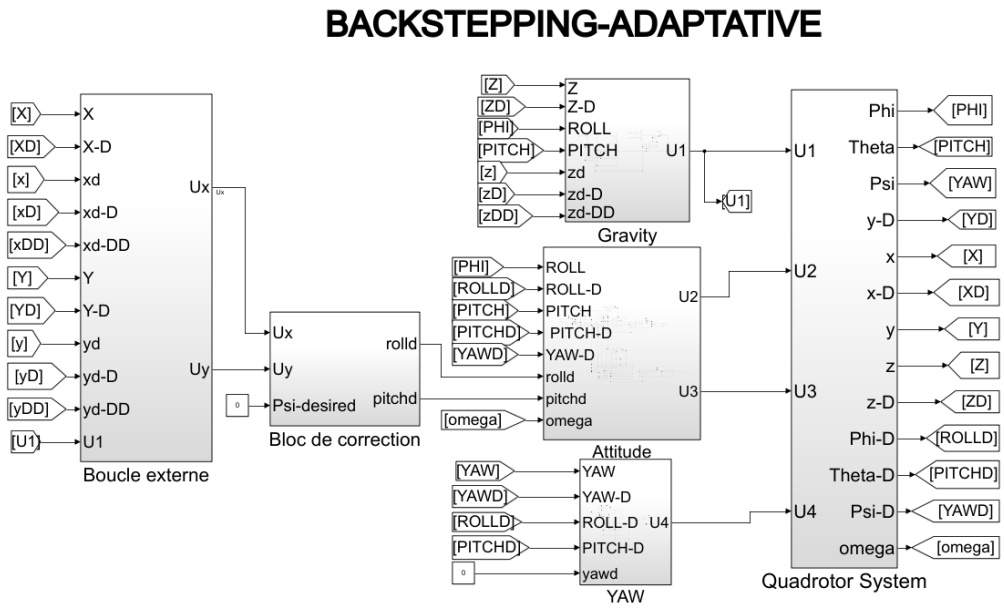

Figure 4. 7 Modèle Simulink pour Backstepping adaptative

La Figure 4.7 présente le modèle Simulink de la commande backstepping adaptative. La même stratégie de commande est appliquée pour cette simulation. La boucle interne contient quatre lois de commande : commande de rotation selon l'axe x par U_2 (3.46), commande de rotation selon l'axe y par U_3 (3.47), commande de rotation selon l'axe z par U_4 (3.48) et la commande de mouvement suivant l'axe z par U_1 (3.49). La boucle externe inclut deux lois de commande : commande de mouvement suivant l'axe x par U_{x} (3.50) et commande de mouvement suivant l'axe y par U_y (3.51). Passant par le bloc de correction, qui contient deux commandes : la commande pour l'angle désirée $\emptyset$ par $\emptyset_d$ (3.32) et la commande pour l'angle désirée θ par θ_d (3.33).

Les entrées du modèle du quadrotor sont : $U_1, U_2, U_3\ et\ U_4$ et les sorties sont les trois position x, y et z et les trois angles de rotations $\emptyset$, θ et ψ.

Le tableau suivant illustre les gains de la commande backstepping-adaptative (γ = 0.00112) :

Tableau 4. 3 Gains du backstepping-adaptative

Désignation	k_{x1}	k_{x2}	k_{x3}	k_{x4}	k_{x5}	k_{x6}	k_{x7}	k_{x8}	k_{x9}	k_{x10}	k_{x11}	k_{x12}
Valeurs	211	179	211	179	201	171	2.7	1.4	291	131	57	57

❖ **Simulation avec une trajectoire d'un cercle :**

Deux simulations ont été faites. La première, en utilisant les paramètres du modèle (Tableau 4.1). Les résultats de cette simulation sont présentés dans la Figure 4.8. La deuxième en modifiant les paramètres du modèle : la masse m, les inerties I_{xx}, I_{yy}, I_{zz} et J_r (10% de réduction). Cette réduction représente la perte des quatre éléments de protection qui protège les hélices du quadrotor. Ces résultats sont présentés dans la Figure 4.9.

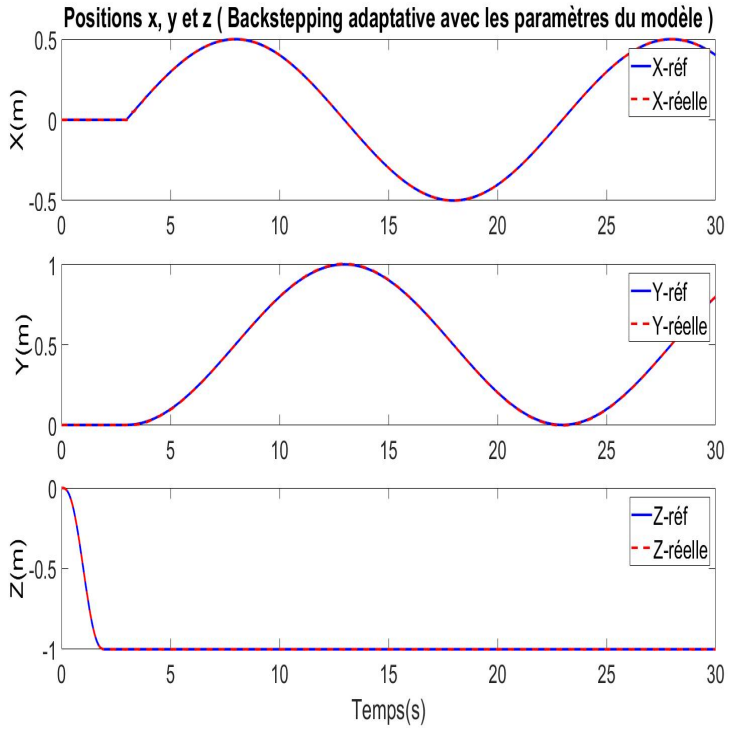

Figure 4. 8 Suivie de trajectoire du cercle pour les positions x, y et z

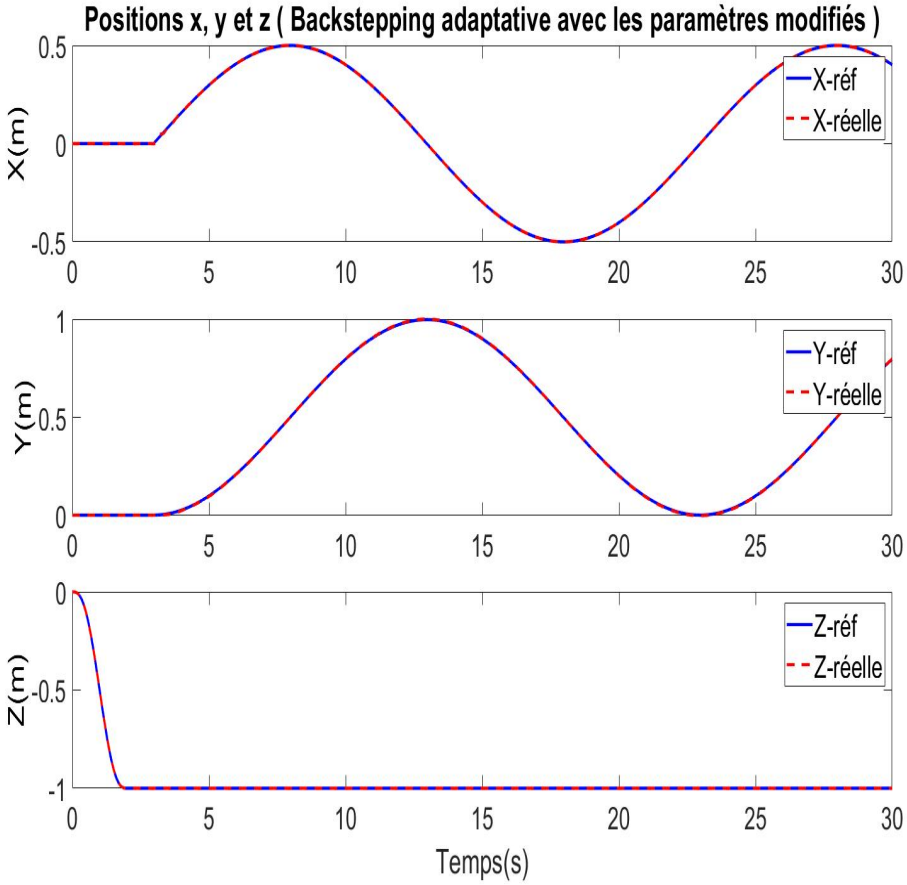

Figure 4. 9 Suivie de trajectoire du cercle pour les positions x, y et z

L'ajout de la loi d'adaptation à la commande Backstepping, a permet de rendre la commande robuste. Les résultats pour les deux simulations, montrent une amélioration de l'erreur selon l'axe x. En effet, nous pouvons conclure que la commande backstepping-adaptative est performante avec une trajectoire de cercle. Ce dernier, est une trajectoire plus complexe dont la dérivée par rapport au temps n'est pas nulle. Le quadrotor converge vers la trajectoire de référence désirée. Les deux figures : Figure 4.10 et Figure 4.11, présentent la réalisation d'une trajectoire cercle en 3D par la commande backstepping-adaptative, en utilisant les paramètres du modèle et les paramètres modifiés.

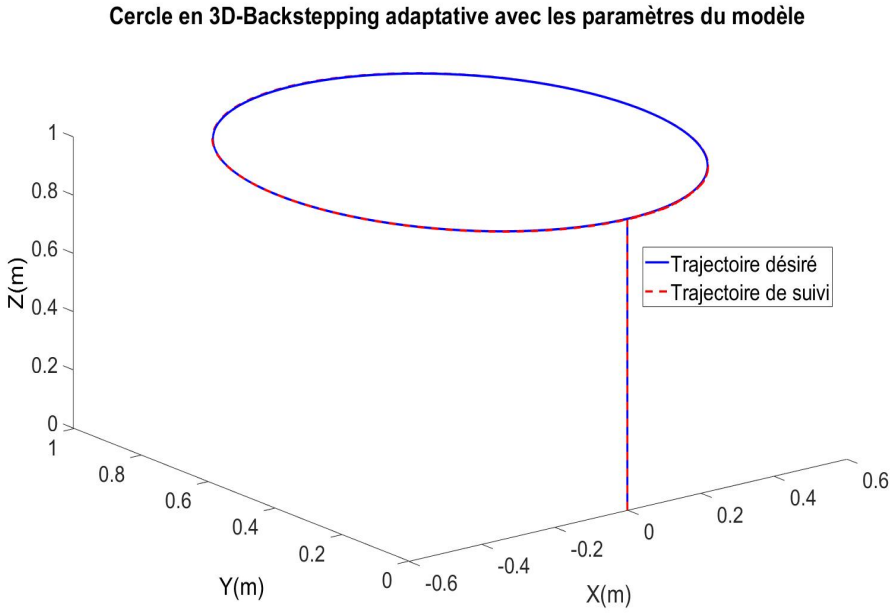

Figure 4. 10 Cercle en 3D-Backstepping-Adaptative avec les paramètres du modèle

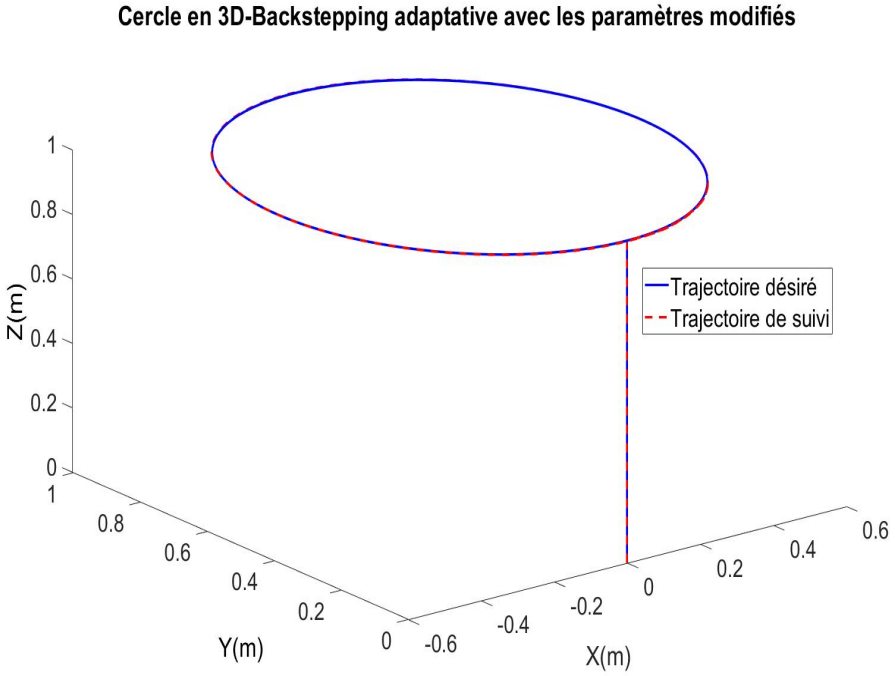

Figure 4. 11 Cercle en 3D-Backstepping-Adaptative avec les paramètres modifiés

4.4 Conclusion

Deux approches de commande ont été validées et testées en effectuant des simulations de suivi d'une trajectoire circulaire. Le premier test est une simulation du modèle dynamique du quadrotor en boucle fermée avec la commande Backstepping. Cette commande garantie la stabilité du quadrotor en boucle fermée et assure une bonne performance de celui-ci avec une petite erreur selon l'axe x. Le deuxième test consiste en une simulation de la commande backstepping adaptative pour assurer de meilleures performances, sans affecter le bon fonctionnement du système, notamment sa stabilité. Cette simulation nous a permis de vérifier que l'ajout de la loi d'adaptation rend la commande plus robuste face aux incertitudes paramétriques.

CHAPITRE V RÉSULTATS EXPÉRIMENTAUX

Introduction

Après la validation de notre contrôleur dans le chapitre précédent et la détermination des gains concevables par les résultats de simulations du modèle sur Matlab/Simulink, les résultats ont montré la performance de la loi de commande Backstepping-adaptative appliquée sur le modèle. Par la suite, nous allons procéder à faire une implémentation réelle pour approuver ces performances par des résultats expérimentaux. Ces derniers permettront de vérifier les performances du système en mode vol stationnaire. Les vérifications expérimentales seront effectuées en utilisant Matlab 2020b et d'autres outils MathWorks. En effet, ce chapitre évoquera en premier lieu le matériel et les logiciels nécessaires pour cette opération. Ensuite, nous présentons les concepts généraux liés à la boîte à outils utilisés pour compiler et envoyer les algorithmes de contrôle de Matlab/Simulink au PARROT Mambo sans fil via Bluetooth par un « dongle CSR 4.0 ». Enfin, les résultats expérimentaux seront relevés.

Le matériel nécessaire pour cette opération est le suivant :

- Le drone Parrot Mambo (contient une batterie chargée)
- Chargeur batterie
- Câble USB
- « Dongle USB Bluetooth » avec un pilote « CSR USB Bluetooth » déjà installé.

Les logiciels nécessaires qui sont installés sont les suivants :

- Matlab R2020b
- Simulink
- Simulink support package for Parrot minidrone

- Aerospace block set
- Simulink Coder
- Simulink 3D Animation

Figure 5. 1 Matériel

La simulation sera effectuée par AsbQuadcopterStart pour accéder, en temps réel, aux données des capteur du quadrator. Toutes ces simulations ont été conçues pour être utilisées sur les drones de la famille des minidrones PARROT (section 5.1). L'un des principaux avantages de l'utilisation des quadrators du minidrone PARROT est la possibilité d'effectuer des tests pratiques sans un investissement élevé.

Pour ce travail, on a utilisé le PARROT Mambo. Ces simulations contiennent l'implémentation des lois de contrôle précédentes qui assurent la stabilité du quadrotor Parrot Mambo lors des tests effectués.

Tout d'abord, une brève introduction de la boîte à outils « Simulink Support Package for Parrot Minidrones » (Section 5.2) et les concepts généraux liés à la commande asbQuadcopterStart (section 5.3) seront présentés. Ensuite, on introduit dans la section 5.4, la plate-forme d'essai qui présentera les étapes de la réalisation des tests expérimentaux et l'acquisition de données. Enfin, dans la section 5.5, seront présentés les résultats obtenus lors des essais.

5.1 PARRAOT Mambo

Les minidrones PARROT sont des systèmes dynamiques ultra-compacts, dotés de quatre hélices qui en font un quadricoptère. Ils peuvent être contrôlés à partir d'un smartphone ou d'une tablette. Ils peuvent être très stables grâce au pilote automatique. Ils peuvent combiner des signaux de gyroscopes à 3 axes et d'accéléromètres à 3 axes, un capteur de pression pour l'altitude de vol, un capteur à ultrasons pour un vol de précision près du sol et une caméra orientée vers le bas pouvant être utilisée pour le traitement du flux optique et des images.

5.2 SIMULINK SUPPORT PACKAGE PARROT mindrone

Le « Toolbox Simulink Support Package for Parrot Minidrones » permet aux utilisateurs du logiciel Matlab/Simulink de mettre en œuvre des algorithmes de contrôle de la trajectoire et de la stabilité ainsi que des algorithmes d'acquisition de données par communication via Matlab/Simulink et des algorithmes de contrôle de la stabilité, ainsi que des algorithmes d'acquisition de données par communication Bluetooth entre le matériel du quadrotor et un ordinateur. Le « dongle » USB 4.0 Bluetooth doit être utilisé pour cette communication, car il a la fonctionnalité de connexion à un réseau ad-hoc de groupe (GN). Les algorithmes implémentés dans le quadrotor grâce à l'utilisation de cette boîte à outils peuvent accéder aux données des capteurs internes du véhicule, tels que des capteurs à ultrasons, des accéléromètres, des gyroscopes, des capteurs de pression atmosphérique et des capteurs de température et caméra.

Pour utiliser la boîte à outils Simulink Support Package for Parrot Minidrones, il est nécessaire d'avoir le logiciel Matlab/Simulink versions 2017a et plus (notre travail sera fait avec la version 2020b).

AsbQuadcopterStart

La simulation AsbQuadcopter a été développée pour simuler l'algorithme de contrôle de la trajectoire et la stabilité avant de l'envoyer au quadrotor. Cette simulation sera

également utilisée pour accéder en temps réel les données des capteurs et de la batterie du quadrotor. Pour lancer la simulation AsbQuadcopter, il suffit de taper "asbQuadcopterStart" dans le champ Fenêtre de commande du logiciel Matlab, puis appuyez sur la touche "Entrée" du clavier.

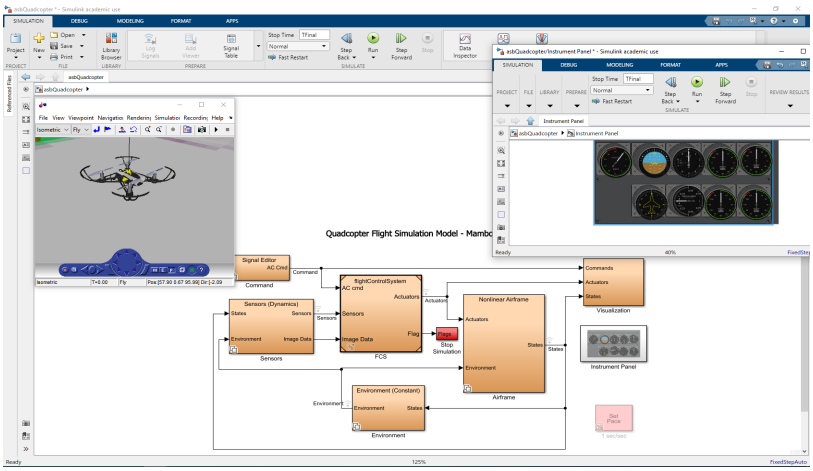

Figure 5. 2 AsbQuadcopterStart

Après avoir exécuté la commande pour ouvrir la simulation AsbQuadcopter comme indiqué dans la Figure 5.2, deux fenêtres s'ouvriront automatiquement. La première consiste en une fenêtre pour les mouvements du quadrotor en trois dimensions où l'utilisateur peut simuler le vol du véhicule aérien avant de l'envoyer au modèle réel. L'environnement 3D a été créé pour la visualisation seulement, et ne peut pas être modifié.

La deuxième fenêtre présente le modèle de simulation sous forme de blocs en utilisant la fonction Outil Simulink. Contrairement au modèle de visualisation décrit ci-dessus, le modèle de simulation peut être édité, ce qui permet aux utilisateurs du logiciel Matlab/Simulink d'implémenter leurs propres algorithmes de contrôle de vol, à partir des modèles mathématiques linéaires et non linéaires qui caractérisent la dynamique de vol des minidrones de la compagnie PARROT.

Le modèle de simulation AsbQuadcopter est divisé en six blocs principaux représentés en orange qui permettent d'arrêter la simulation (Bloc : "Stop Simulation") et de contrôler la vitesse de simulation (Bloc : "Set Pace"), comme le montre la Figure 5.3. Chacun des blocs principaux seront brièvement détaillés ci-dessous.

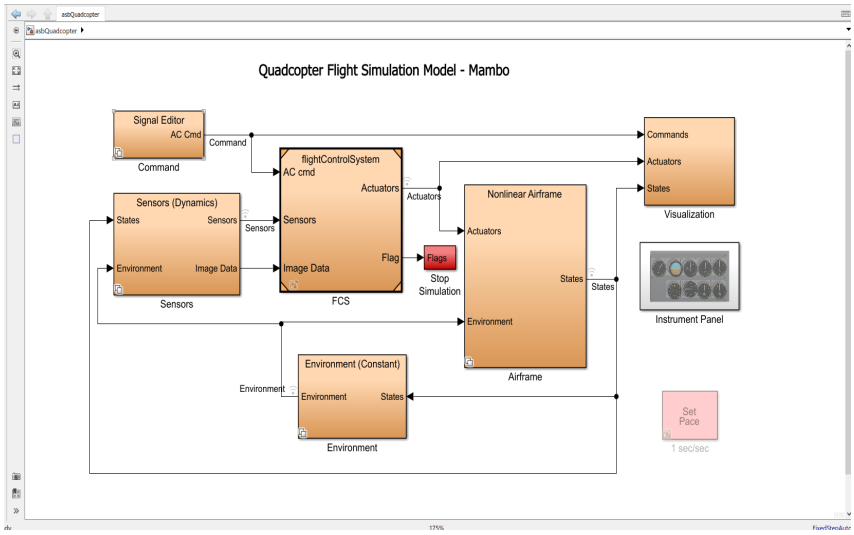

Figure 5. 3 Modèle de simulation (Quadcopter Flight Simulation Model)

Bloc Airfame

Le bloc "Airfame" contient les représentations mathématiques régissant le vol du quadrotor. Ce bloc est divisé en deux sous-systèmes qui se réfèrent à la modélisation linéaire et non linéaire (nous utilisons le modèle non linéaire).

Bloc FCS

Le bloc "FCS" est responsable du contrôle du vol, dans lequel les codes de vol sont générés qui sont envoyés au quadrotor. Ce bloc possède un sous-système de contrôle appelé FlightControlSystem, où nous avons implémenté nos contrôleurs basés sur le développement d'une technique backstepping adaptative.

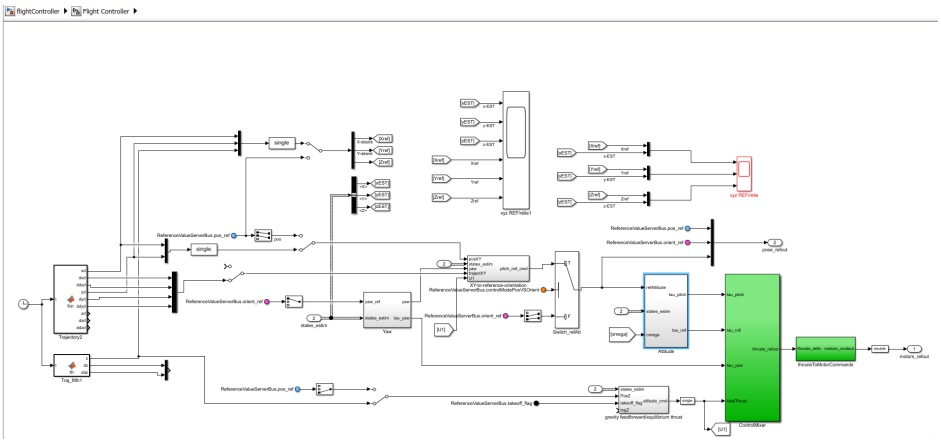

Figure 5. 4 Sous system flight Controller

Dans ce sous-système, ce qui nous intéresse est le contrôleur appelé FlightControl qui est divisé en quatre blocs de contrôle principaux représentés dans Figure 5.4.

Bloc Commande

Le bloc "Commande" est utilisé pour fournir une entrée au système, où cette entrée se réfère à un signal de position en fonction du temps, caractérisé comme étant la trajectoire à parcourir par le quadrotor. Ce signal de trajectoire est constitué de six composantes, et est utilisé comme référence dans les comparateurs utilisés dans les boucles de contrôle de position. Le bloc "Commande" est divisé en quatre sous-systèmes comme spécifiés ci-dessous :

- Sous-système "Signal Build" : Utilisé lorsque l'utilisateur veut implémenter des signaux d'entrée (trajectoire) sous sa forme continue dans le domaine temporel.

- Sous-système "Joystick" : utilisé lorsque l'utilisateur veut mettre en œuvre des signaux d'entrée (trajectoire) en utilisant une commande par joystick.

- Sous-système de données "Spreadsheet Data" : utilisé pour mettre en œuvre des signaux d'entrée provenant de feuilles de calcul à l'aide de l'outil Excel.

- Sous-système "Data" : utilisé lorsque l'utilisateur veut mettre en œuvre des signaux d'entrée (trajectoire) sous forme vectorielle.

Bloc Capteurs

Le bloc "Capteurs" permet de simuler la dynamique et le bruit des capteurs utilisés dans le quadrotor

Bloc Environnent

Le bloc "Environnement" est utilisé pour modéliser le vecteur gravité, la pression atmosphérique et les capteurs. Il est divisé en deux sous-systèmes : Environnement constant et Environnement variable qui dépend de la position.

Bloc Visualisation

Le bloc "Visualisation" est utilisé pour tracer les signaux de sortie du système, et déclencher la fonction de visualisation 3D pour afficher les mouvements du quadrotor générés par la simulation Asbquadcopter.

5.3 Plateforme d’essais

Les essais expérimentaux ont été réalisés dans le laboratoire des systèmes de contrôle de l'Université du Québec en Abitibi-Témiscamingue (UQAT). La plate-forme est composée d'un drone Parrot Mambo, un dongle USB Bluetooth 4.0, un chargeur batterie, un câble USB et un ordinateur avec tous les logiciels nécessaires installés. La figure 5.5 ci-dessous présente une photo de la plate-forme d'essais utilisée pour effectuer les tests expérimentaux.

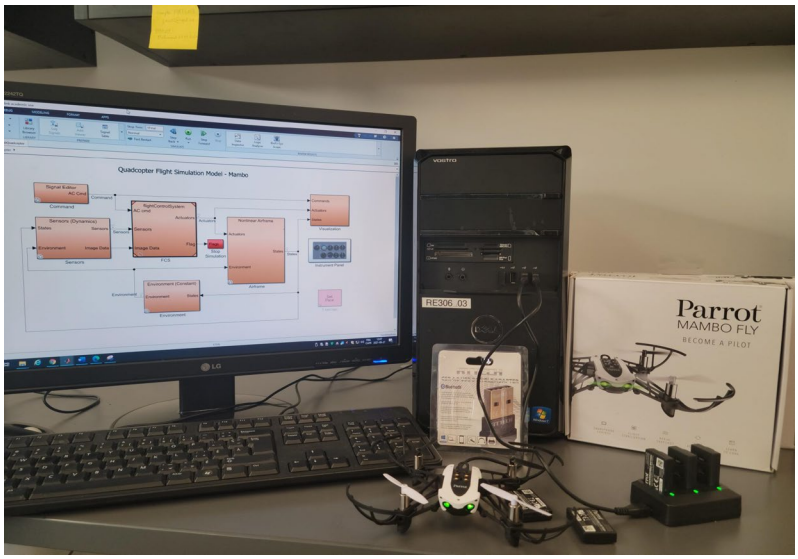

Figure 5. 5 Plate-forme d'essais

À partir du projet, AsbQuadcopterStart, et après avoir créé notre contrôleur Backstepping Adaptative dans le bloc FCS, le codeur Simulink est utilisé pour générer et exécuter le code à partir du modèle Simulink. Le code source généré peut être utilisé pour des applications de simulation et d'implémentation en temps réel aussi, y compris l'accélération de la simulation, le prototypage rapide et les tests de matériel. On peut régler et surveiller le code généré à l'aide de Simulink ou exécuter et interagir avec le code en dehors de MATLAB et Simulink. Ce codeur Simulink permet d'enregistrer les données de vol sur le quadrator et d'accéder au code généré à partir des modèles Simulink. Le code généré est ensuite déployé sur Parrot Mambo sans fil par la communication Bluetooth.

En utilisant le **Simulink Support Package for Parrot Minidrones,** l'algorithme de contrôle sera envoyé et déployer via Bluetooth. Le dongle CSR 4.0 est chargé d'établir une connexion Bluetooth entre l'ordinateur et le Parrot Mambo. Les instructions étape par étape pour établir cette communication sont décrites dans l'Annexe.

Le flux de travail de programmation et de test est résumé dans la Figure suivante.

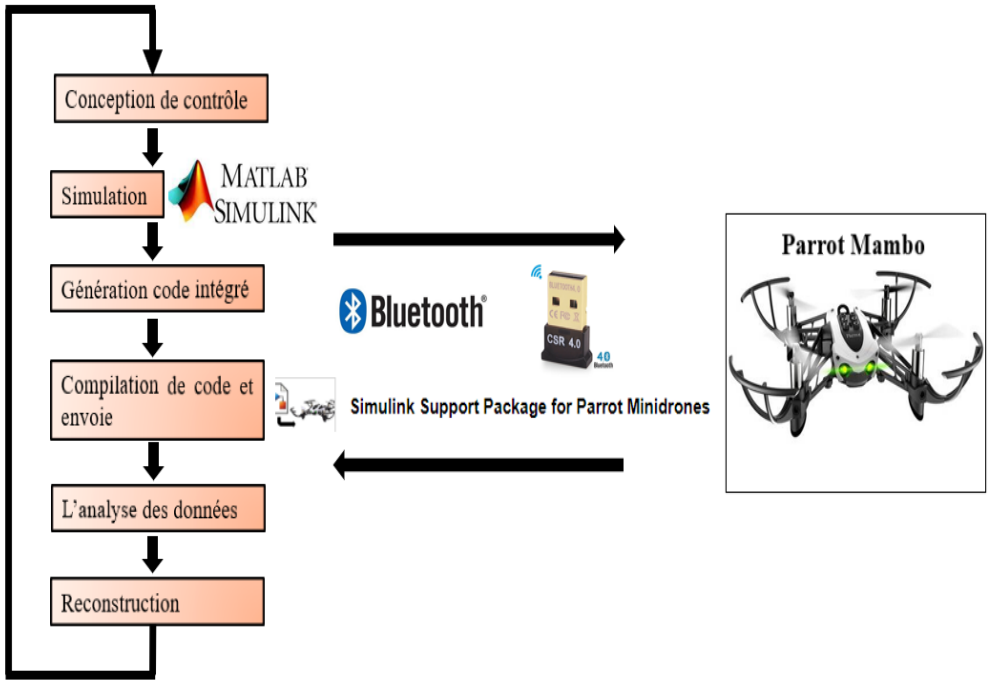

Figure 5. 6 Architecture d'implémentation du Parrot Mambo

Pour assurer la sécurité du matériel et du personnel, il est possible d'énumérer d'autres directives importantes : l'espace où les tests de vol sont effectués doit être grand, au moins 20 pieds par 20 pieds par 10 pieds de hauteur pour éviter d'endommager le matériel (Parrot Mambo), l'environnement et tout autre observateur. Au cours des essais, on a remarqué que les petites pièces et les matériaux spécifiques au sol (comme un tapis par exemple) pourraient causer des problèmes de stabilité de vol en raison de rebondissement ou de l'absorption des signaux ultrasonores.

5.4 Résultats et discussions

Les données expérimentales sont obtenues en temps réel à partir de la mémoire intégrée dans le Parrot Mambo qui est conçue pour enregistrer les données essentielles telles que les capteurs, les positions (x, y, z) et les rotations (roll, tangage, lacet). Initialement, l'altitude souhaitée est fixée à 1.5 mètre, tandis que pour l'angle de rotation, le lacet est fixé à zéro.

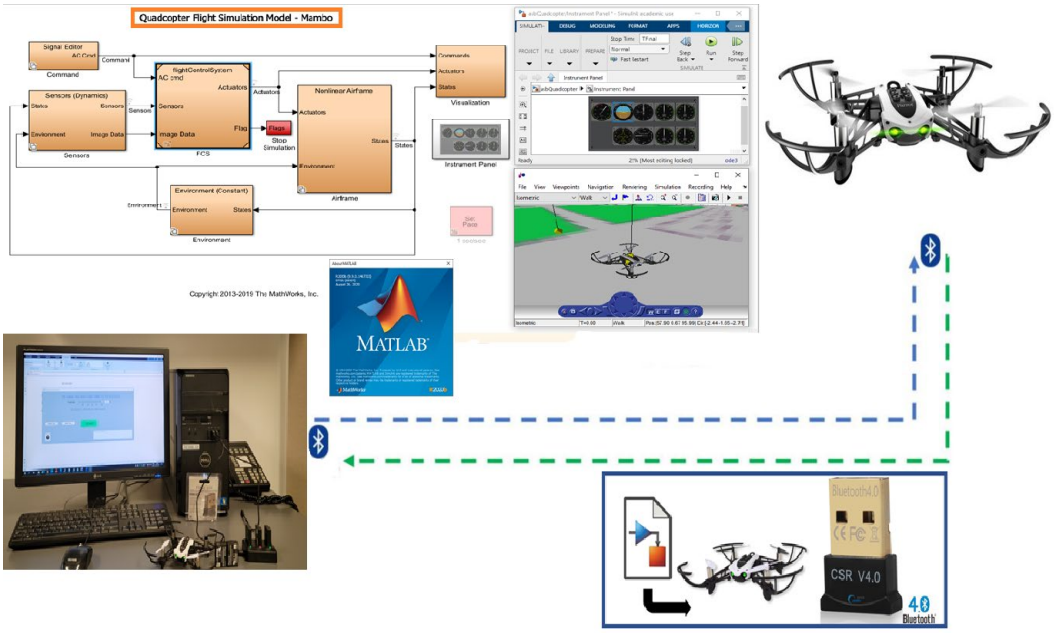

Figure 5. 7 Configuration expérimentale

Les performances de la méthode de Backstepping appliquée sur le système sont démontrées expérimentalement via la plateforme d'essais du Parrot Mambo. La mise en œuvre pratique présenté par la Figure 5.7 est basée sur le Simulink support package for PARROT mini drone. Dans les tests, on a été utilisé une batterie Lithium-Ion Polymère (Li-Po) de la marque Shoot, modèle XT-412, ayant une autonomie de 6 à 8 minutes

5.5.1 Résultats du test expérimental de vol stationnaire à une altitude de 1.5 mètre

La figure ci-dessous montre les résultats de simulation lors de l'implémentation de la commande proposée, sur le modèle asbQuadcopter. Cette commande garantie les performances de stabilisation du Parrot Mambo à 1.5 m. Les résultats de cette commande pour un vol stationnaire de 1.5 mètre, présentés dans la Figure 5.8. Ces résultats sont très satisfaisants pour la stabilisation de position et d'altitude. Les erreurs de position sont présentées dans la Figure 5.9. L'erreur est de l'ordre 4×10^{-3} mètre.

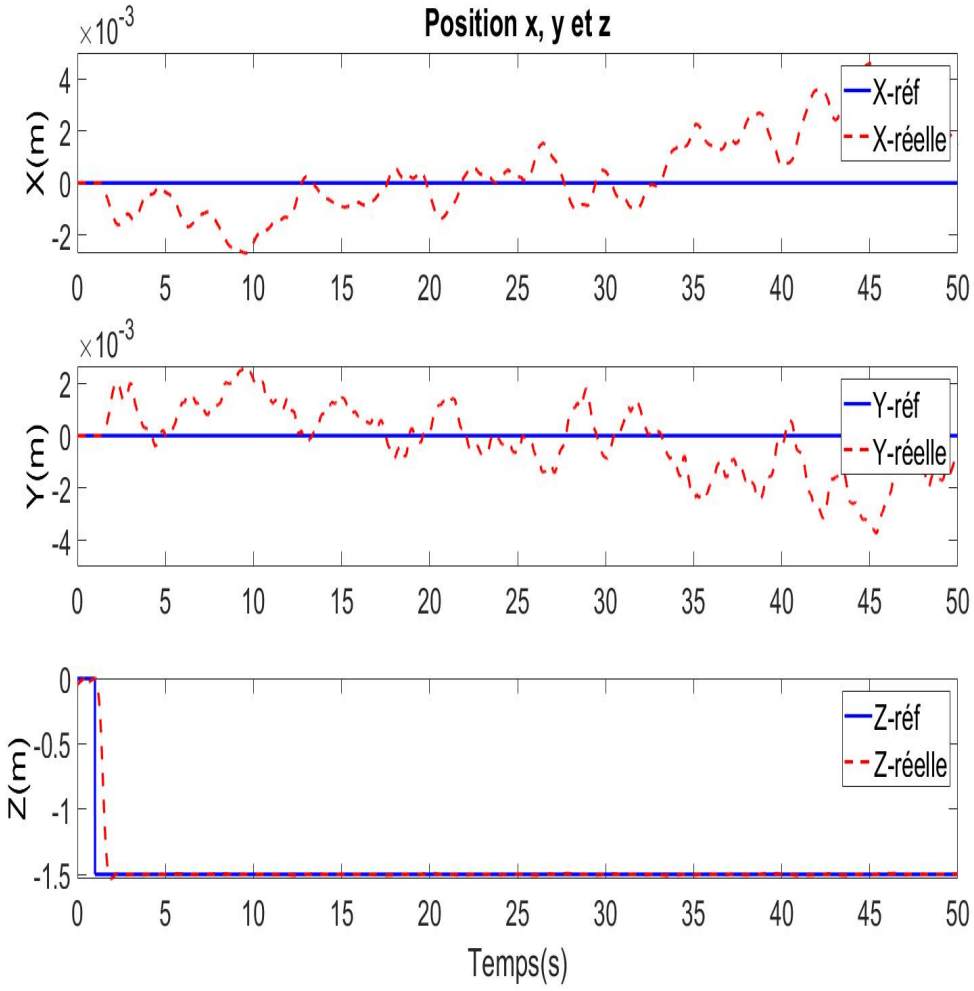

Figure 5. 8 Position x, y et z : vol stationnaire

Le Parrot Mambo se met en vol stationnaire à une altitude de 1,5 mètre pendant une durée de 6 minutes. On a abouti de contrôler parfaitement l'altitude selon l'axe z et de garantir les performances de suivi de la trajectoire d'attitude, bien que l'erreur de mouvement de translation selon l'axe x et y est de l'ordre 4×10^{-3} mètre.

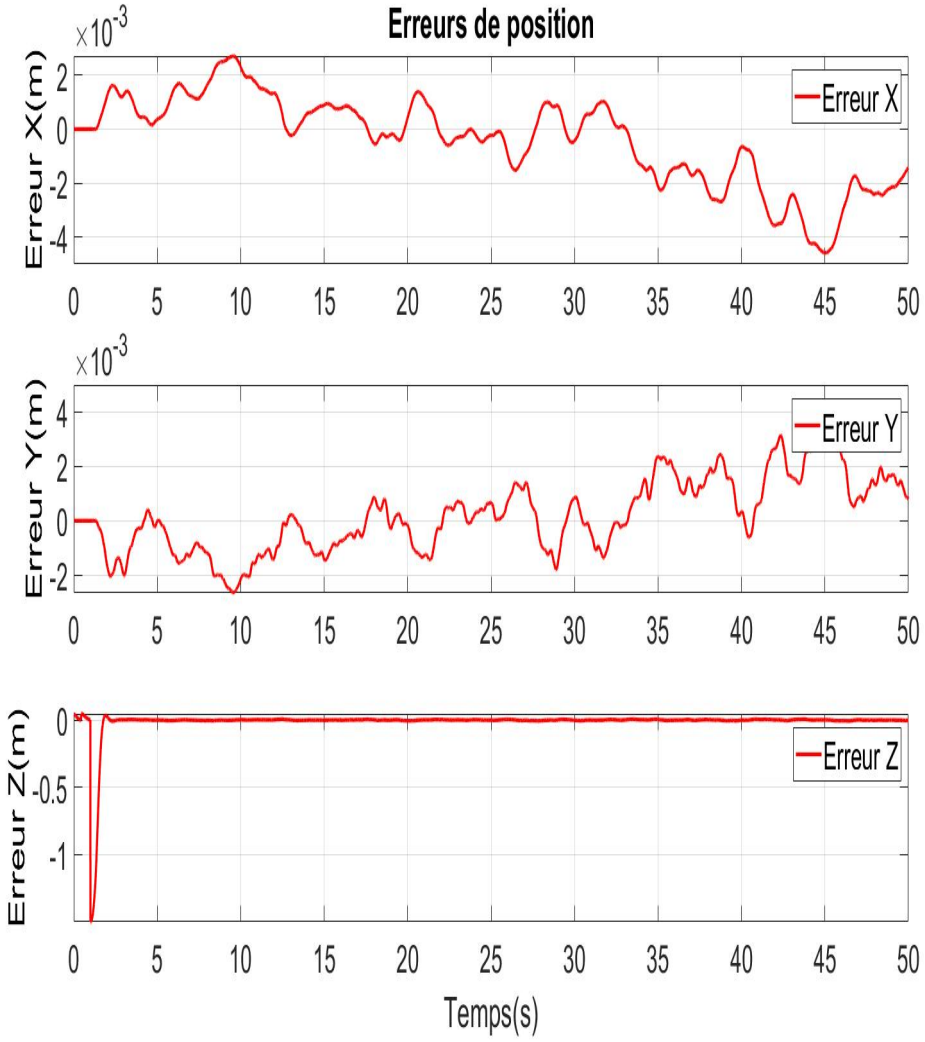

Figure 5. 9 Erreurs de position selon x, y et z : vol stationnaire

5.5.2 Résultats du test expérimental de vol de trajectoire cercle

La Figure 5.10 présente le suivi de trajectoire d'un cercle dans l'espace à un hauteur de 1 mètre effectuée par le Parrot Mambo. C'est bien clair que la commande robuste force le système à suivre la trajectoire désirée malgré les petites erreurs selon les axes x et y Les résultats de position de x, y et z, sont présentées dans la Figure 5.10. Les erreurs sont illustrées à la Figure 5.11. On voit que le système de contrôle par backstepping est

capable de stabiliser solidement le Parrot Mambo et le déplacer sur la trajectoire désirée. Les performances de la commande proposée, en suivi de trajectoire ainsi que les faibles valeurs d'erreurs, sont de bons indicateurs de la bonne performance de la commande proposée. L'erreur de position selon l'axe x est de l'ordre 0.13 mètre et selon l'axe y est de l'ordre 0.07 mètre

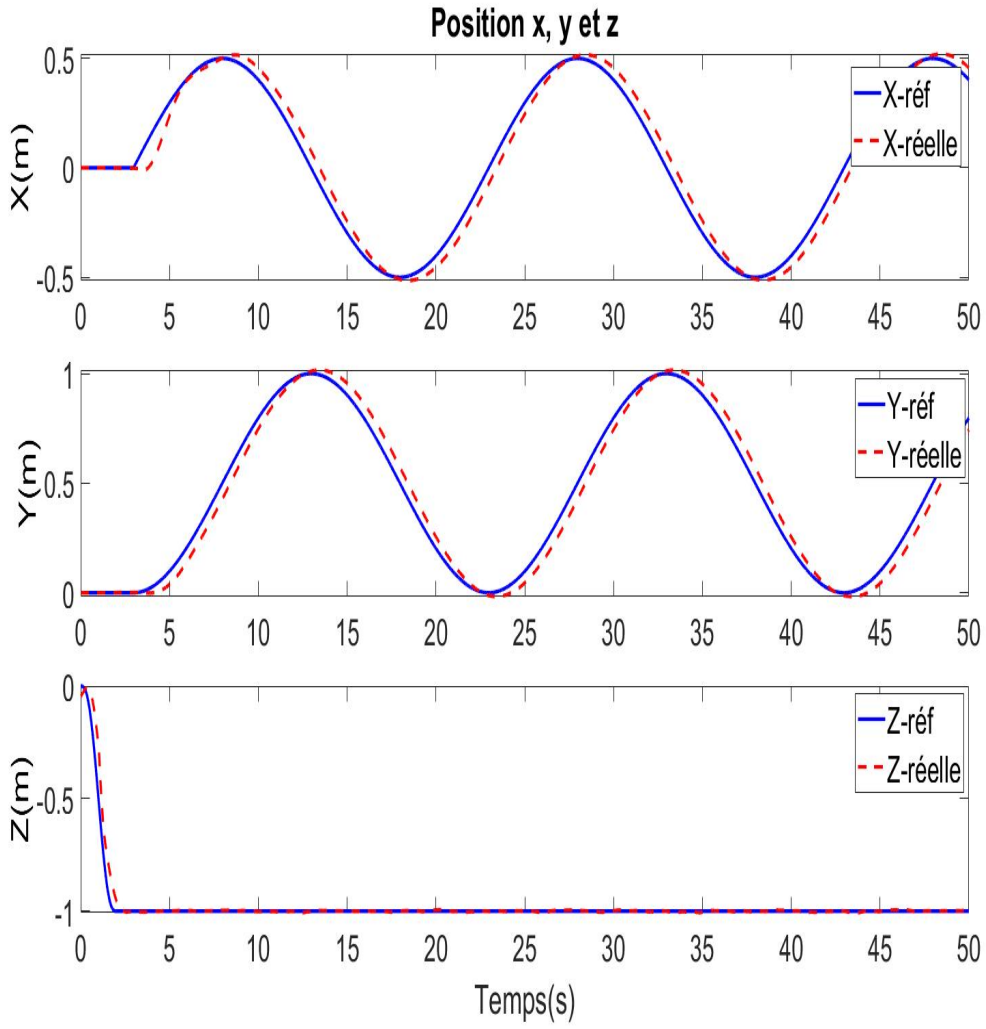

Figure 5. 10 Position x, y et z : Trajectoire cercle

Le système de contrôle est capable de stabiliser solidement le Parrot Mambo et le déplacer sur la trajectoire désirée. Les performances de la commande, proposée en suivi

de trajectoire et les faibles valeurs d'erreurs, sont de bons indicateurs de l'exactitude de la commande proposée. Les erreurs de positions sont illustrées à la Figure 5.11, sont de l'ordre 0.13 de selon x et 0.07 selon y avec un suivie parfait selon l'axe z. En effet, nous pouvons interpréter que l'origine de cette erreur provient principalement des perturbations et par les mesures des capteurs.

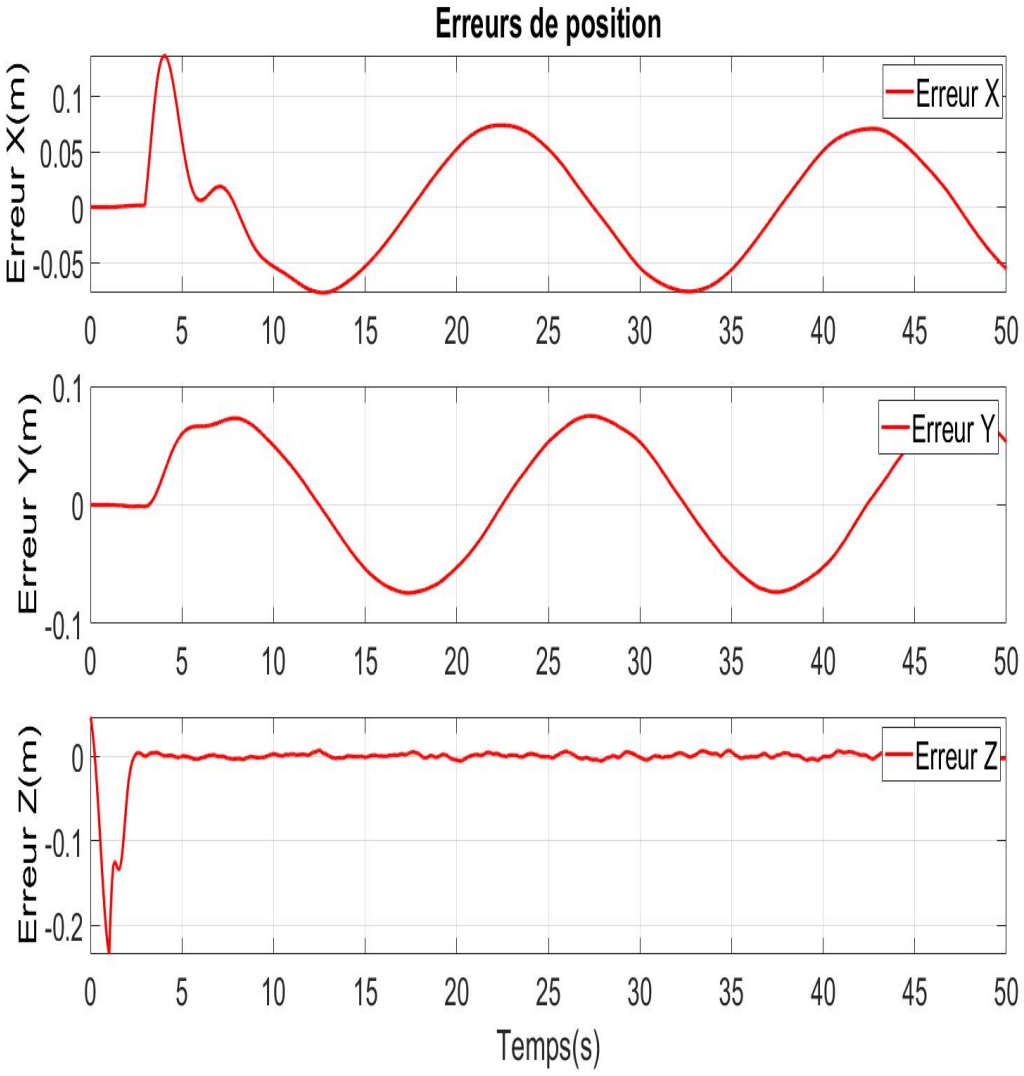

Figure 5. 11 Erreurs de position selon x, y et z : Trajectoire cercle

5.5 Conclusion

Dans ce chapitre, nous avons implémenté notre approche de commande sur le matériel Parrot Mambo et nous avons validé notre travail par des résultats expérimentaux avec plusieurs essais. En effet, nous avons démontré dans un premier temps la performance de notre commande pour le vol stationnaire à une altitude de 1,5 mètre surtout pour le contrôle de la position et d'altitude. Ensuite, nous avons validé la performance de notre contrôleur par une deuxième trajectoire plus complexe variante dans le temps et dont les dérivées des positions désirées par rapport au temps ne sont pas nulles, c'est une trajectoire d'un cercle dans l'espace à hauteur de 1 mètre. Les résultats ont démontré une bonne stabilité en vol stationnaire et que la commande non linéaire proposée est efficace d'assure le contrôle parfait l'altitude et garantir les performances de suivi de la trajectoire d'altitude du Parrot Mambo en pratique.

CONCLUSION GÉNÉRALE

Le quadrotor est un système mécanique non linéaire. Sa dynamique hautement couplée rend la modélisation précise de ce type d'appareils télécommandés difficile à obtenir et à concevoir. Une telle modélisation, par le formalisme de Newton-Euler, facilite la tâche pour trouver une solution pour commander le système avec une meilleure stabilisation.

Une commande robuste, par backstepping adaptative, a été développée dans le but d'améliorer la qualité du contrôle non linéaire et d'éviter les défauts de stabilité et l'incertitude des paramètres qui se produisent dans le système dynamique.

Par la suite, des simulations du modèle, conçue à l'aide du logiciel Matlab/Simulink pour valider notre contrôleur par une trajectoire non stationnaire dont la dérivée par rapport au temps n'est pas nulle. En effet, les résultats ont montré la performance de la commande en boucle fermée avec cette trajectoire. Le quadrotor converge vers la trajectoire de référence désirée.

Une implémentation sur le matériel a été faite à l'aide du logiciel Matlab/Simulink et d'un dongle Bluetooth CSR 4.0 pour la communication avec le drone Parrot Mambo pour tester et analyser les performances du système. La commande développée sur le modèle du quadrotor fournit de bons résultats en temps réel pour la stabilité et le contrôle d'altitude du système avec le minimum d'erreur.

RECOMMENDATIONS

Bien que le travail présenté dans ce mémoire a permis d'atteindre nos objectifs principaux, soit l'amélioration de performance de vol d'une drone Parrot Mambo en utilisant la commande backstepping adaptative, certains aspects auraient pu être améliorés. En effet, l'approche de commande proposée reste performante au niveau théorique en simulation avec Matlab/Simulink et en pratique pour vols stationnaire et circulaire.

Dans les travaux futurs, il est recommandé d'étudier certains points afin d'améliorer les performances du système.

Premièrement, les erreurs de mesures des différents capteurs, l'incertitude des paramètres et les caractéristiques aérodynamiques qui changent en fonction de l'altitude affectent les performances de la commande. Les effets environnementaux et le facteur de vieillissement jouent un rôle supplémentaire dans le changement des paramètres de la commande. Citons à titre d'exemple les perturbations météorologiques telles que la perturbation du vent et le frottement de l'air et le couplage dynamique du modèle. Ce type particulier d'aéronef est relativement de petite taille et est considéré comme très sensible à la perturbation.

Deuxièmement, la disponibilité d'une chambre à essai. Notons que plusieurs essais doivent être faites au niveau pratique pour différentes trajectoires désirées, mais à cause de l'espace restreint et le manque d'équipement, les tests ont été limités : l'espace où les tests de vol seront effectués doit être grand, au moins 20 pieds par 20 pieds par 10 pieds de hauteur avec un matériel spécifique au sol pour l'absorption des chocs sans cassé le drone en raison de rebondissement ou de l'absorption des signaux ultrasonores.

ANNEXE –CONFIGURATION ET COMMUNICATION PAR BLUETOOTH

Après avoir téléchargé le Toolbox Simulink Support Package pour Parrot Minidrones, en connectant le Parrot Mambo avec le logiciel Matlab/Simulink à travers le dongle USB Bluetooth, les prochaines étapes à réaliser consistent à configurer et à envoyer les algorithmes de vol, en plus de l'acquisition des données des capteurs et de la batterie à partir de la communication Bluetooth entre le logiciel Matlab/Simulink et Parrot Mambo.

Cette section a pour but de présenter toutes les étapes nécessaires pour configurer et envoyer la simulation AsbQuadcopter à Parrot Mambo, et obtenir l'accès aux données de vol, à la décharge de la batterie et aux relevés des capteurs. La configuration pour envoyer la simulation et l'acquisition de données entre le logiciel Matlab/Simulink et le matériel Parrot Mambo sont présentées étape par étape et détaillées ci-dessous.

À partir de la fenêtre principale de MATLAB, ouvrez le gestionnaire " Add Ons ", puis choisissez " Menage Add Ons " et lancez la procédure en cliquant sur l'icône en forme d'engrenage Simulink Support Package for Parrot Minidrones.

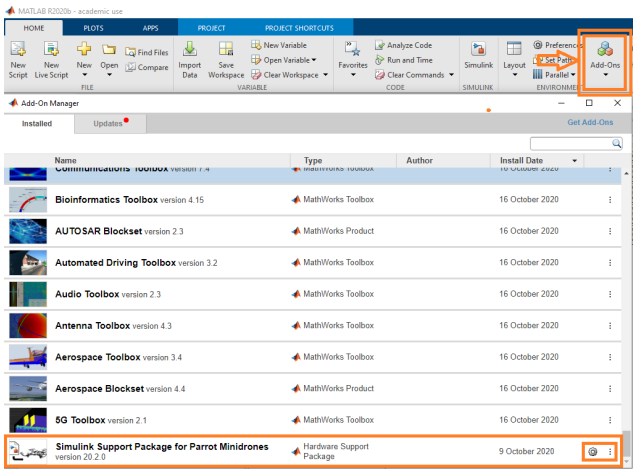

Figure Annexe. 1 Étape de lancer le Toolbox

Connectez le drone avec le câble USB et attendez quelques instants qu'il soit reconnu par le PC. Cliquez sur Suivant et attendez la reconnaissance du drone.

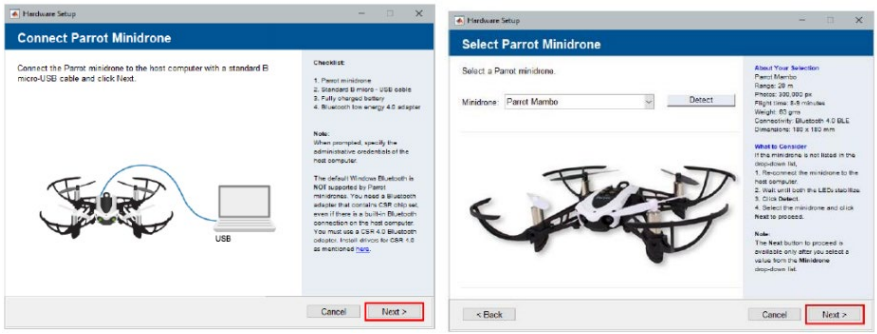

Figure Annexe. 2 Reconnaissance du drone par le câble USB

Appuyez à nouveau sur Next, le fichier du firmware sera chargé dans la mémoire du drone. Il est nécessaire de déconnecter le câble USB du Parrot Mambo pour commencer la procédure d'installation. Pendant l'installation, les LEDs de la face avant clignoteront en orange. Une fois l'installation terminée, les LEDs clignoteront en vert (assurez-vous que ce dernier état se produit pendant au moins 10 secondes pour poursuivre la configuration).

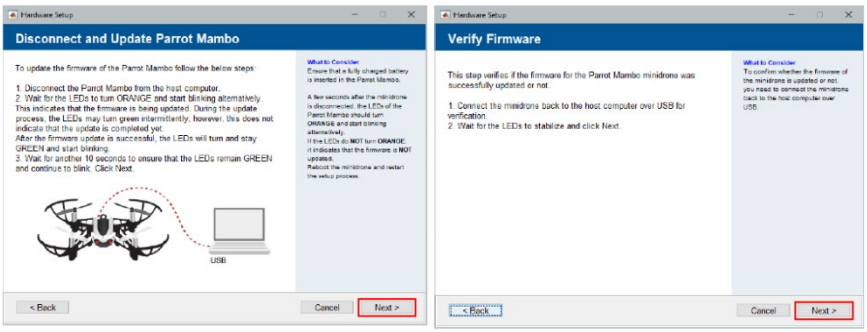

Figure Annexe. 3 Configuration du logiciel

Reconnectez le câble USB au drone (attendez quelques instants qu'il soit reconnu) et procédez en cliquant sur Suivant .

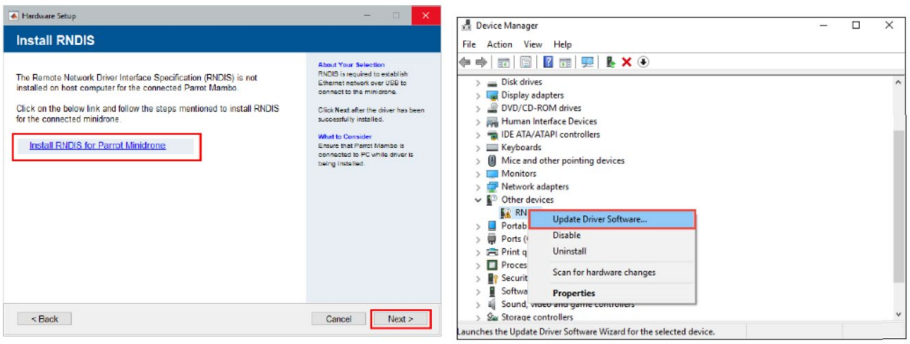

Figure Annexe. 4 Configuration du pilote RNDIS (1)

Avant de poursuivre, il faut suivre la procédure suivante pour installer le pilote RNDIS. Ouvrez le "Gestionnaire de périphériques" et cliquez sur "Autres périphériques". Dans la liste, sélectionnez "RNDIS" et "Update Driver Software...".

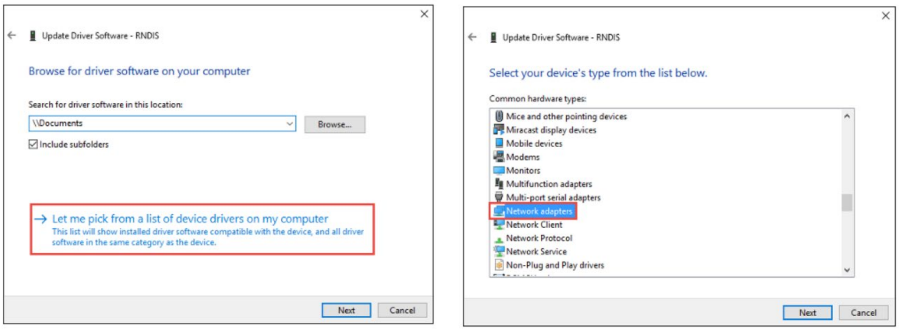

Figure Annexe. 5 Configuration du pilote RNDIS (2)

Dans le nouvel écran, il faut choisir le logiciel pilote dans la liste des périphériques. Parmi les "Types de matériel ", sélectionnez "Adaptateurs réseau".

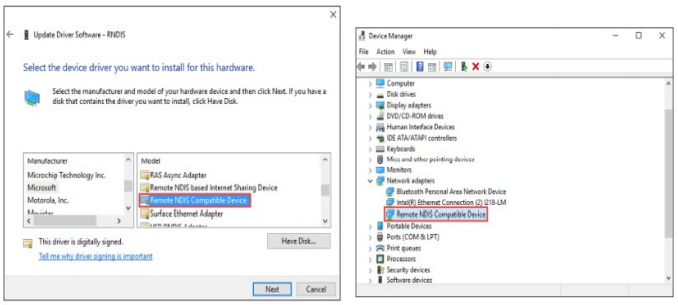

Figure Annexe. 6 Configuration du pilote RNDIS (3)

Recherchez "Microsoft" parmi les fabricants, "Remote NDIS Compatible Device" parmi les modèles. Dans la boîte de dialogue "Update Driver Warning", sélectionnez "Yes". Une fois installé, le périphérique RNDIS peut être trouvé parmi les périphériques dans "Network adapters". Il est maintenant possible de poursuivre la procédure d'installation.

En cliquant sur Next une nouvelle fenêtre s'affiche, contenant des instructions pour connecter le drone via Bluetooth. Tout d'abord, vous devez débrancher à nouveau le câble USB du drone.

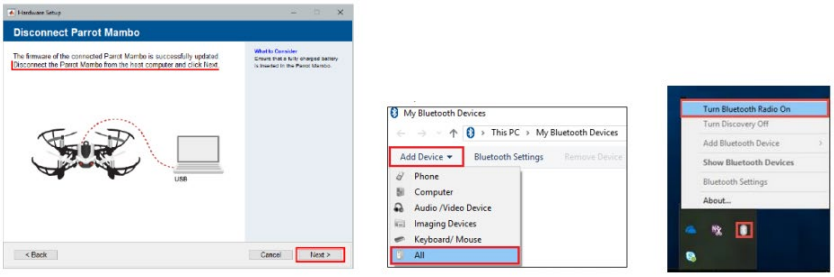

Figure Annexe. 7 Configuration du dispositif radio Bluetooth

Maintenant, dans la fenêtre de l'explorateur de fichiers, cliquez sur "Ce PC", puis double-cliquez sur "Mes périphériques Bluetooth". Ensuite, ouvrez le menu déroulant "Ajouter un périphérique" et sélectionnez "Tous".

La recherche de périphériques Bluetooth commence. Le nom du périphérique à connecter s'appelle "Mambo" suivi de plusieurs chiffres (relatifs au numéro de série du matériel). Il est nécessaire de choisir celui qui a l'icône d'un Joystick ; sinon, la communication entre MATLAB et le drone ne sera pas possible.

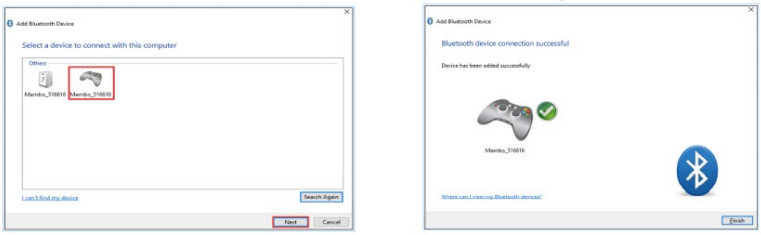

Figure Annexe. 8 Couplage Bluetooth

Une fois couplage terminé, double-cliquez sur le périphérique qui vient d'être ajouté et après avoir appuyé sur le texte de droite au-dessus de l'icône accompagnée du nom Personal Area Networking (NAP), cliquez sur connecter.

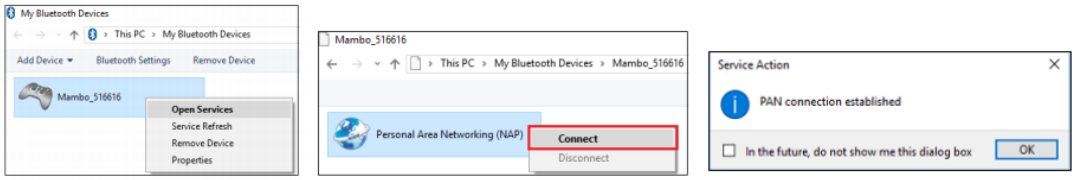

Figure Annexe. 9 Configuration de la connexion PAN

Maintenant, appuyez sur Next, et dans l'écran suivant, cliquez sur "Test connection" pour effectuer un ping afin de vérifier que la communication se produit

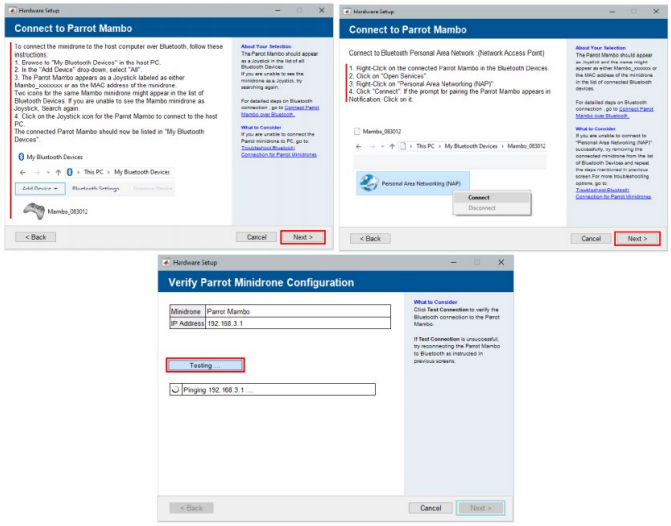

Figure Annexe. 10 Test de connexion Bluetooth

Une fois l'installation terminée et réussie, cliquez sur Suivant, puis sur Terminer pour terminer l'installation. Cette procédure ne doit être effectuée qu'une seule fois. Si nous voulons connecter le drone à d'autres moments, il faut le faire via Bluetooth avec PAN (Personal Area Network).

RÉFÉRENCES

Alqaisi, W. K., et al. (2019). Adaptive sliding mode control based on RBF neural network approximation for quadrotor. 2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE), IEEE.

Alqaisi, W. K. A. (2019). Nonlinear control and perturbation compensation in UAV quadrotor, École de technologie supérieure.

Altug, E. (2003). "Vision based control of unmanned aerial vehicles with applications to an autonomous four-rotor helicopter, quadrotor."

Benallegue, A., et al. (2008). "High-order sliding-mode observer for a quadrotor UAV." International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal **18**(4-5): 427-440.

Benaskeur, A. (2000). Aspects de l'application du Backstepping adaptatif à la commande des systèmes non linéaires, Ph. D, Université Laval, Québec.

Besnard, L., et al. (2012). "Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer." Journal of the Franklin Institute **349**(2): 658-684.

Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying, Epfl.

Bouabdallah, S., et al. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), IEEE.

Bouabdallah, S. and R. Siegwart (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. Proceedings of the 2005 IEEE international conference on robotics and automation, IEEE.

Bouadi, H., et al. (2007). "Modelling and Stabilizing Control Laws Design Based on Sliding Mode for an UAV Type-Quadrotor." Engineering Letters **15**(2): 342-347.

Bouchoucha, M., et al. (2008). Step by step robust nonlinear PI for attitude stabilisation of a four-rotor mini-aircraft. 2008 16th Mediterranean Conference on Control and Automation, IEEE.

Boudguiga, O. (2016). Commande à saturation pour le contrôle de la position d'un robot volant de type quadrotor, École de technologie supérieure.

Bresciani, T. (2008). "Modelling, identification and control of a quadrotor helicopter." MSc Theses.

Castillo, P., et al. (2004). "Real-time stabilization and tracking of a four-rotor mini rotorcraft." IEEE Transactions on control systems technology **12**(4): 510-516.

Castillo, P., et al. (2004). Stabilization of a mini-rotorcraft having four rotors. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), IEEE.

Choi, I. and H. Bang (2014). "Quadrotor-tracking controller design using adaptive dynamic feedback-linearization method." Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **228**(12): 2329-2342.

Das, A., et al. (2009). "Backstepping approach for controlling a quadrotor using lagrange form dynamics." Journal of Intelligent and Robotic Systems **56**(1): 127-151.

Dolatabadi, S. H. and M. J. Yazdanpanah (2015). MIMO sliding mode and backstepping control for a quadrotor UAV. 2015 23rd Iranian Conference on Electrical Engineering, IEEE.

Du, H., et al. (2017). "Finite-time formation control for a group of quadrotor aircraft." Aerospace Science and Technology **69**: 609-616.

Fang, Z. and W. Gao (2011). Adaptive integral backstepping control of a micro-quadrotor. 2011 2nd International conference on intelligent control and information processing, IEEE.

Ghandour, J., et al. (2014). Feedback linearization approach for standard and fault tolerant control: Application to a quadrotor UAV testbed. Journal of Physics: Conference Series, IOP Publishing.

Hamel, T., et al. (2002). "Dynamic modelling and configuration stabilization for an X4-flyer." IFAC Proceedings Volumes **35**(1): 217-222.

Lee, D., et al. (2009). "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter." International Journal of control, Automation and systems **7**(3): 419-428.

Madani, T. and A. Benallegue (2006). Control of a quadrotor mini-helicopter via full state backstepping technique. Proceedings of the 45th IEEE Conference on Decision and Control, IEEE.

Madani, T. and A. Benallegue (2007). Backstepping control with exact 2-sliding mode estimation for a quadrotor unmanned aerial vehicle. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE.

McGeoch, D., et al. (2005). MIMO sliding mode attitude command flight control system for a helicopter. AIAA Guidance, Navigation, and Control Conference and Exhibit.

Mian, A. A., et al. (2008). "Backstepping based PID control strategy for an underactuated aerial robot." IFAC Proceedings Volumes **41**(2): 15636-15641.

Mistler, V., et al. (2001). Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. Proceedings 10th IEEE international workshop on robot and human interactive communication. Roman 2001 (Cat. no. 01th8591), IEEE.

Mohd Basri, M. A., et al. (2015). "Intelligent adaptive backstepping control for MIMO uncertain non-linear quadrotor helicopter systems." Transactions of the Institute of Measurement and Control **37**(3): 345-361.

Mokhtari, A. and A. Benallegue (2004). Dynamic feedback controller of Euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, IEEE.

Oloo, J. O. and S. I. Kamau (2018). Quadcopter Control Algorithms in the event of Loss of One of the Actuators: A Review. Proceedings of Sustainable Research and Innovation Conference.

Park, J., et al. (2014). "Landing site searching and selection algorithm development using vision system and its application to quadrotor." IEEE Transactions on control systems technology **23**(2): 488-503.

Rudolph, J. and E. Delaleau (1998). "Some examples and remarks on quasi-static feedback of generalized states." Automatica **34**(8): 993-999.

Sumantri, B., et al. (2013). Least square based sliding mode control for a quad-rotor helicopter. Proceedings of the 2013 IEEE/SICE International Symposium on System Integration, IEEE.

Wang, L. and H. Jia (2014). "The trajectory tracking problem of quadrotor UAV: Global stability analysis and control design based on the cascade theory." Asian Journal of Control **16**(2): 574-588.

Xu, R. and U. Ozguner (2006). Sliding mode control of a quadrotor helicopter. Proceedings of the 45th IEEE Conference on Decision and Control, IEEE.

Yali, Y. and W. Yuanxi (2012). "Controller design of quadrotor aerial robot." Physics Procedia **33**: 1254-1260.

Yu, Y., et al. (2015). Robust backstepping tracking control of uncertain MIMO nonlinear systems with application to quadrotor UAVs. 2015 IEEE International Conference on Information and Automation, IEEE.

Zhou, Q.-L., et al. (2010). Design of feedback linearization control and reconfigurable control allocation with application to a quadrotor UAV. 2010 Conference on Control and Fault-Tolerant Systems (SysTol), IEEE.