Représentations d'état des systèmes linéaires - BE

Institut Polytechnique des Sciences Avancées (IPSA)

Simulation d'un drone

Réalisé par :

Florian Alaux, Lefol Romain, Guilhem Perret-Bardou

Encadrants :

M. Jacob

Année académique : 2024-2025

06 juin 2025

Table des matières

Introduction

Dans le cadre d'une commande passée par notre client, M.Jacob, nous avons été mandatés pour réaliser l'étude et la simulation du drone professionnel Parrot Anafi USA à l'aide de MATLAB et Simulink.
Ce document présente l'avancement de ce projet.

L'objectif opérationnel est de développer un environnement de simulation exploitable pour reproduire le comportement dynamique du drone dans des scénarios de vol définis, en intégrant des contraintes physiques

Le simulateur que nous développons doit donc permettre la validation de commandes (PID, stabilisation et guidage) et la visualisation des comportements en vol pour permettre une analyse précise des données. A partir de cela, il sera possible d'évaluer les performances de ce drone sous contraintes dynamiques réelles pouvant replacer des expérimentations physiques pouvant endommager ce drone particulièrement coûteux, pouvant aller de 8400€ à 16800€.
Notre note technique est structurée par cas d'étude, correspondant aux étapes suivantes :

— **Commande Hover** : stabilisation initiale en assurant un vol stationnaire par compensation de la gravité.

— **Commande d'une pose** : mise en place d'une commande en boucle fermée permettant au drone de se déplacer d'une position initiale A vers une position cible B.

— **Limitations du modèle et tuning PID** : intégration de limitations mécaniques du drone (poussée maximale, angles admissibles, saturation) et ajustement des correcteurs PID en conséquence pour obtenir des performances cohérentes.

— **Suivi d'une trajectoire** : évaluation de notre drone sur le suivi d'un parcours défini et présentation de l'ensemble des résultats nous permettant de suivre les écarts, ses performances et son temps sur le parcours.

Nous allons présenter dans chaque section les hypothèses que nous avons retenues, les résultats obtenus et nos analyses critiques.

TP1 : La commande "hover"

Cette première phase a pour objectif de se familiariser avec la modélisation du quadrirotor, en se concentrant à placer le drone à l'équilibre, puis nous ferons varier certains paramètres afin de comprendre son comportement.

À ce stade, la seule commande transmise au drone est la poussée individuelle de chaque moteur. Ces poussées sont ajustées de manière à compenser exactement la force gravitationnelle, maintenant ainsi le drone à une altitude constante, dans une position dite de « vol stationnaire » (*hover*). Bien que cette approche soit simplifiée et peu représentative d'un système réel où les commandes ne sont généralement pas appliquées directement aux moteurs, mais transmises via des composants mécaniques et électroniques intermédiaires elle constitue une base pertinente pour débuter l'étude.

Par ailleurs, plusieurs hypothèses simplificatrices sont faites : les vitesses sont supposées atteintes instantanément et les forces extérieures se limitent au seul poids du drone. Malgré ces approximations, ce modèle initial est un bon point de départ.

Nous avons d'abord étudié le système théorique. L'application du principe fondamental de la dynamique ainsi que du théorème du moment cinétique nous a permis de déterminer les six équations du mouvement, correspondant aux six degrés de liberté du drone. À partir de celles-ci, nous avons construit les matrices A, B, C et D de la représentation d'état, en les remplissant avec les termes issus de notre modélisation. Comprendre ces matrices est essentiel au bon déroulement du projet, en effet se sont elles qui vont définir les équations du mouvement et faire le lien entre les positions en (x,y,z) et les angles (θ, ϕ, ψ) qui correspondent respectivement à l'angle du tangage, du roulis et du lacet.

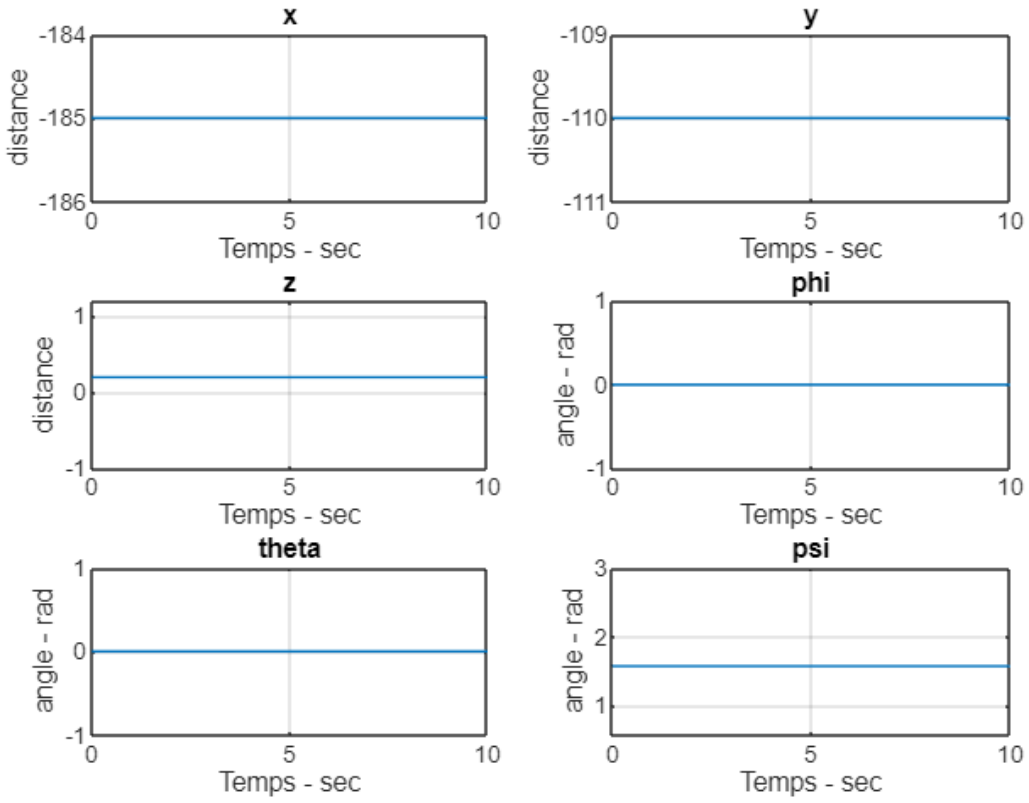

Figure 1 – Variables en Hover

Nous voyons sur les graphiques que la position en vol stationnaire est bien respectée et que nos variable reste constante au cours du temps aux valeurs initiales données.

Afin de valider l'initialisation du comportement de notre drone, nous cherchons ensuite à comprendre son comportement en faisant varier les entrées initiales, toujours en boucle ouverte (les entrées initiales déterminent entièrement le mouvement du drone).

— **Vitesses initiales** $(\dot{x}, \dot{y}, \dot{z})$ **non nulles** : Nous remarquons qu'ajouter en entrée des vitesses non nulles $(\dot{x}, \dot{y}, \dot{z})$ n'aura aucune influence concernant nos angles (θ, ϕ, ψ), en effet le drone se contentera d'avancer à vitesse constante faisant ainsi augmenter linéairement la position sur chaque axe et à l'infini puisque, dans notre simulation, le drone n'est pas freiné par les frottements de l'air.

— **Angle** (ψ) **non nuls** : L'ajout d'un angle de lacet initial non nul ne provoque pas de variation des autres paramètre.

— **Angles** (θ, ϕ) **non nuls** : Cela devient intéréssant lorsque nous décidons de mettre des vitesses ($\dot{x}, \dot{y}, \dot{z}$) nuls mais des angles ($\theta, \phi$) non nuls.
Nous observons qu'un angle de tangage initiale à $\frac{\pi}{4}$ aura pour effet de déplacé le drone selon l'axe x tout en gardant (y, z, ϕ, ψ) constant.

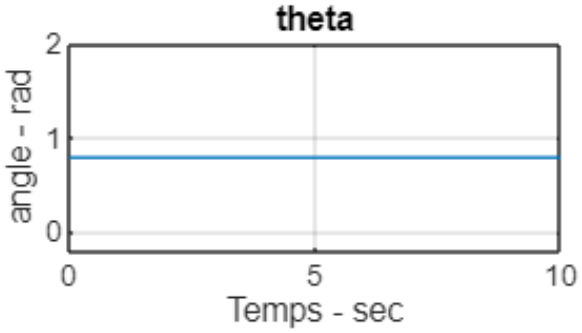

FIGURE 2 – Evolution angle theta

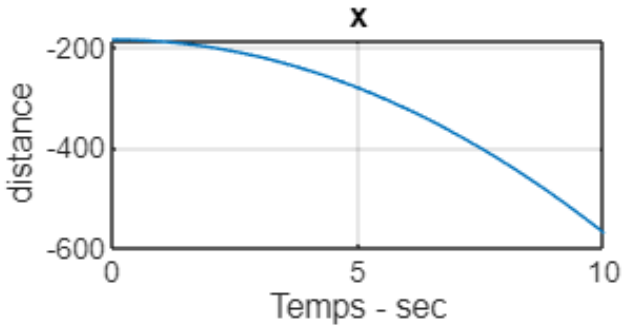

FIGURE 3 – Evolution position x

Ce phénomène s'éxplique par un changement d'orientation de la force des 4 moteurs, en effet celle-ci ne sera plus orienté uniquement selon l'axe z (comme pour la position "hoover") mais aura une composante selon l'axe x. Nous retrouvons la même analogie concernant le roulis et la position selon y.
Il est très important de noté que sans l'approximation des petits angles, une augmentation de ces deux angles devraient provoquer une baisse de l'altitude puisque la poussé totale selon z à diminué mais le poids lui n'a pas changé.

Après avoir posé les bases de notre modélisation, intéressons nous à l'asservissement de notre drone et au montage Simulink associé.

TP2 : Commande d'une "pose"

Dans la première partie, nous avons établi la représentation d'état du système et fixé une position initiale en appliquant des commandes constantes directement aux moteurs, afin de compenser la gravité.
Nous souhaitons à présent faire évoluer notre modèle Simulink de manière à ne plus spécifier les commandes au niveau des moteurs, mais à définir directement une consigne sur la position en translation et l'orientation en lacet. Cette consigne sera exprimée sous la forme d'un vecteur

$$[z, y, x, \psi]$$

que nous désignerons comme la pose du drone.
L'objectif est de déplacer le drone d'une pose stable P_A, correspondant à une position A sur la carte, vers une autre pose stable P_B en un point B.
Cette section décrit la démarche mise en œuvre pour concevoir le modèle Simulink permettant d'atteindre cette transition, ainsi que les résultats obtenus lors du déplacement du drone entre les deux poses.
L'étude des actions des quatre moteurs du drone, ainsi que de leur influence sur ses mouvements, a permis de montrer qu'il est possible de découpler ces mouvements des actions des moteurs. Cette relation peut être exprimée sous forme matricielle :

$$\mathbf{F} = \left(\frac{M^{-1}}{2}\right)\mathbf{C}$$

où :

- $\mathbf{F}$ est le vecteur des forces générées par chaque moteur,
- $\mathbf{C}$ est le vecteur de commande : $\mathbf{C} = [T_c, \phi_c, \theta_c, \psi_c]^T$,
- $\mathbf{M}$ est une matrice de transformation reliant les commandes aux forces moteurs.

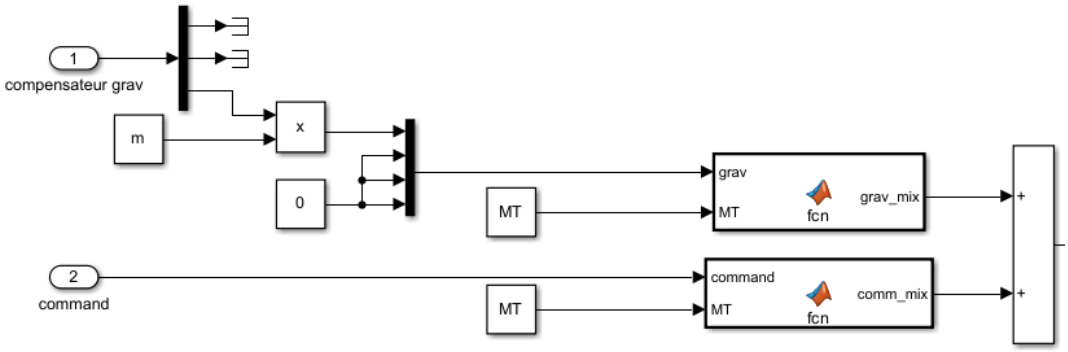

FIGURE 4 – Extrait simulink du compensateur de gravité et des motor mixing

Dans ce modèle, l'effet de la gravité (considéré comme constant) est traité séparément de la commande dynamique $\mathbf{C}$, qui varie au cours du temps. La formulation matricielle ci-dessus permet de répartir la force totale requise entre les différents moteurs, afin de produire le mouvement souhaité.
Par ailleurs, les moteurs étant commandés en vitesse de rotation, il est nécessaire d'établir une relation entre la poussée générée et la vitesse de rotation des hélices. Pour cela, nous faisons l'hypothèse simplificatrice suivante :

$$F_{M_i} = k_t \cdot \omega_{M_i}^2$$

où :

- $\mathbf{F_{M_i}}$ est la force générée par le moteur i,
- $\omega_{\mathbf{M_i}}$ est la vitesse de rotation de l'hélice du moteur i,
- $\mathbf{k_t}$ est un coefficient de poussée caractéristique du moteur et de l'hélice.

Ces vitesses de rotation sont ensuite transmises au système, qui les reconvertit en forces dans notre représentation d'état (établie au TD1). Bien que la transformation des forces moteurs en vitesses, puis à nouveau en forces, ne soit pas immédiatement utile, elle deviendra nécessaire lorsque nous introduirons les limitations imposées par le constructeur sur la vitesse de rotation maximale des hélices.

Note but est de pouvoir guider le drone vers la pose cible $[T, \varphi, \theta, \psi]$. Pour cela, nous adaptons dynamiquement la commande en fonction de l'erreur entre la position réelle actuelle et la cible B, à l'aide d'un contrôleur en boucle fermée.

Comme nous l'avons vu précédemment au TP1, l'angle θ influe négativement sur la position x et de la même façon, φ influe positivement sur la position y du drone. Ainsi, l'erreur en y est liée à la commande en φ, tandis que l'erreur en x détermine celle en θ, avec un signe opposé. Nous avons donc deux correcteurs proportionnels dérivés pour ajuster ces commandes continuellement, sans oublier le bloc multiply -1 devant l'erreur en x.

Pour ce qui est de l'erreur en z, elle pilote directement la poussée T, donc l'ensemble des commandes $[T, \varphi, \theta, \psi]$ est suivi à chaque instant par des correcteurs proportionnels dérivés. Cela nous assure la stabilité de notre drone et sa précision tout le long du déplacement.

Une fois que nous avons implémenté cela dans le modèle Simulink qui nous avions précédemment, nous pouvons enfin tester une simulation de trajet en commençant par définir une pose stable P_A comme état initial du système et une seconde pose P_B étant la destination où le drone doit aussi se stabiliser.
Pour cette simulation, nous avons choisi les poses suivantes :

$$P_A = \begin{bmatrix} 0 \text{ m} \\ 0 \text{ m} \\ z_{ground} \\ 0 \end{bmatrix} \quad P_B = \begin{bmatrix} 120 \text{ m} \\ 8 \text{ m} \\ z_{ground} + 4 \text{ m} \\ -\frac{\pi}{4} \end{bmatrix}$$

Nous avons choisi cette arrivée car cela correspond aux coordonnées devant le parc de la carte et l'angle ψ oriente la caméra vers celui-ci.

Nous avons donc un déplacement de :
- 120 m en direction x,
- 8 m en direction y,
- 4 m en altitude (z),
- ainsi qu'une rotation en lacet (yaw) de $-\frac{\pi}{4}$ radiant

Avec ces distances, nous avons un test suffisamment poussé pour vérifier la capacité du drone à effectuer simultanément des déplacements en translation selon les trois axes et une rotation autour de l'axe vertical. Il constitue un cas d'usage représentatif pour valider le bon fonctionnement de notre système de commande et de stabilisation.

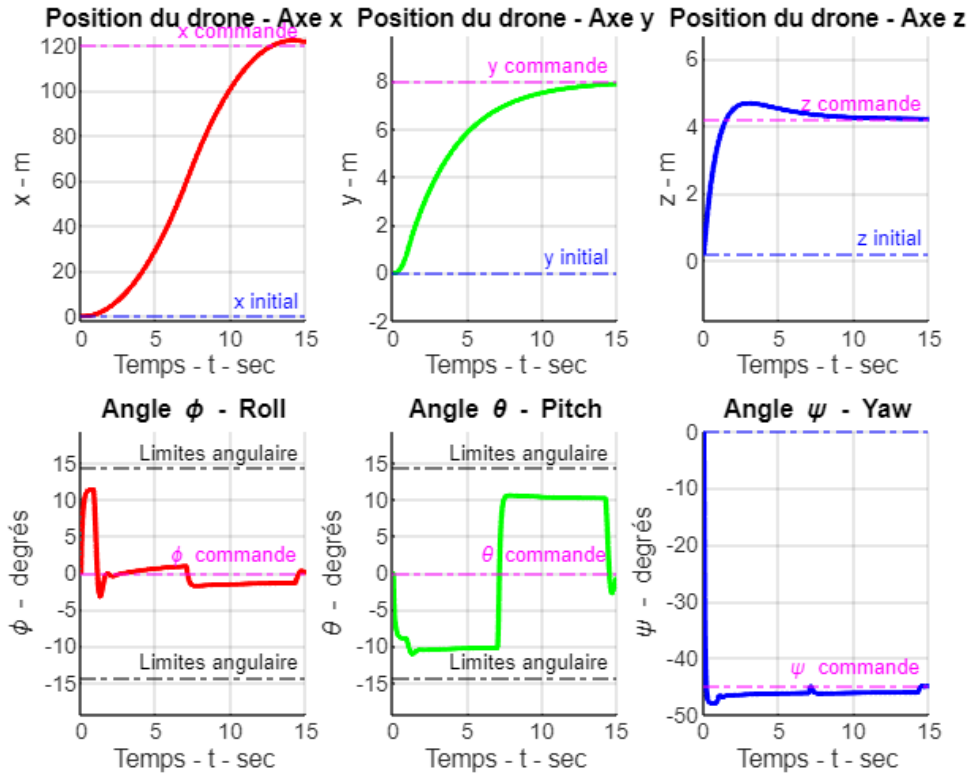

FIGURE 5 – Evolutions des différentes positions et angles du drone en fonction du temps (s)

Avec ces graphiques, nous pouvons remarquer que les variations de position selon les trois axes x, y et z se déroulent de manière relativement progressive et fluide. Toutefois, nous pouvons remarquer un léger dépassement de la valeur cible sur la position x et z. Malgré cela, le drone parvient à se stabiliser à l'altitude demandée après quelques instants.
Ce dépassement est principalement dû à une vitesse de montée excessive lors de la phase de transition, entraînant une inertie plus difficile. Sur l'axe x, c'est explicable par la plus longue distance à parcourir qui laisse le temps au drone d'accélérer, pour le dépassement en z par contre, nous ne pouvons pas émettre cette même hypothèse étant, avec ses $4m$, le déplacement le plus court. Mais cela montre tout de même qu'une trop grande vitesse est atteinte lors de la montée qui n'arrive pas à être freinée totalement à temps et laisse ce dépassement.
Avec cette légère correction à faire, le drone finit par converger vers la position souhaitée en environ quinze secondes.

Si nous nous intéressons maintenant aux angles durant ce trajet, nous pouvons remarquer des courbes très "rectangulaires". Les transitions d'un angle à l'autre se font très rapidement, presque instantanément donnant ces courbes aux variations droites et verticales. Mais nous pouvons remarquer sur chacune d'elle qu'un dépassement de la commande est aussi présent.Encore une fois, l'erreur angulaire est corrigée pour converger vers la commande à la fin du trajet.

Nous pouvons donc conclure grâce à ces graphiques que nos contrôleurs corrigent bien les erreurs et ajuste les mouvements du drone pour qu'il finisse sa course à la position attendue. Mais pour le moment, notre modèle est entièrement libre sur la puissance à utiliser. Cela explique les mouvement quasi instantanés sur les angles où le décollage très rapide. Nous allons voir dans la partie suivante comment faire en sorte que notre modèle corresponde au drone réel que nous devons simuler.

TP3 : Limitations du modèle et tuning PID

Dans cette troisième partie, l'objectif est d'identifier précisément les caractéristiques du drone que nous emploierons, afin d'évaluer ses limites physiques et de concevoir nos régulateurs en conséquence. Nous devons donc modéliser un PARROT Anafi USA, un drone professionnel à application militaire. La ligne directrice que nous suivrons est d'obtenir un drone typé "race" très réactif et dynamique. Les données constructeur que nous exploitons sont notamment ses dimensions, la poussée maximale des moteurs, notée $T_{\max}$ qui vaut 28 newtons, soit 7 newtons par moteur, sa masse de 500g ce qui est particulièrement léger pour ses performances, ainsi que la vitesse de rotation maximale de $1152 rad/s$. Il est aussi dit que ce drone a une limite de déplacement angulaire de 300°/s ce que nous avons trouvé assez élevé et nous amènera quelques difficultés plus tard. Pour implémenter cela, nous effectuerons différents ajustements sur nos correcteurs PD et PID et nous ajouterons des saturations afin d'obtenir des déplacements cohérents avec ce dont ce drone est capable dans la réalité. Mais commençons tout d'abord par déterminer les limites des commandes.

PID petite boucle

Afin de couvrir toutes les situations, même extrêmes, prenons la consigne la plus contraignante que l'on puisse imposer à notre drone, c'est-à-dire les valeurs maximales autorisées pour chaque angle définies par :

$$[\varepsilon_{\varphi_{\max}}, \varepsilon_{\theta_{\max}}, \varepsilon_{\psi_{\max}}] = [0.5, 0.5, \pi]$$

Dans un premier temps, on ≪ déconnecte ≫ la grande boucle en envoyant en entrée les valeurs limites pour chaque angle en tant que constantes. Afin de fixer notre consigne limite pour la poussée maximale, nous souhaitons déterminer la marge de poussée nécessaire pour exécuter ces ordres sans forcer le drone au-delà de ses capacités. La première étape consiste à observer les forces résultantes associées à T_c, φ_c, θ_c et ψ_c dans la petite boucle en introduisant les valeurs de saturation ci-dessus.

Lors de l'ajustement des PID, nous nous sommes appuyés sur l'outil de réglage intégré dans *Matlab*, en étudiant simultanément la réponse à un échelon et l'effort généré par le contrôleur. Ce dernier indicateur nous renseigne sur la quantité d'énergie que le système doit fournir pour suivre une consigne unitaire.
L'un des objectifs principaux était de limiter cet effort de commande afin de rester dans les capacités physiques des moteurs, qui atteignent rapidement leur seuil de saturation. En effectuant plusieurs ajustements, nous avons remarqué que l'intensité de l'effort du contrôleur était liée à la marge de fonctionnement de chaque axe. Cette relation nous a permis de mieux orienter les réglages pour atteindre un comportement satisfaisant du système. C'est-à-dire un comportement sans débordement de valeur cible (pour le premier cas la valeur cible sera la valeur de saturation) ainsi qu'une certaine rapidité puisque notre modèle est typé "race".

Par ailleurs, pour garantir une réponse rapide sans compromettre la stabilité, nous avons dû ajuster deux paramètres essentiels qui sont la marge de phase ainsi que la bande passante. L'ensemble de cette phase de réglage s'est donc basé sur une démarche progressive, où chaque itération nous rapprochait d'un compromis acceptable entre performance et stabilité malgré l'attrait de notre drone.

En appliquant les consignes angulaires maximales précédemment définies, nous effectuons une simulation en utilisant les formules :

$$\text{marge}_{\max} = \max(|T_\theta|) + \max(|T_\phi|) + \max(|T_\psi|)$$

et

$$T_{\text{cmax}} = T_{\max} - mg - \text{marge}_{\max}$$

Nous obtenons via Matlab, une $\text{marge}_{\max}$ d'environ 14 N et T_{cmax} d'environ 9N. Pour finir, la commande maximale est donc de $C_{\max} = [T_{\text{cmax}}, 0.5, 0.5, \pi]$.
La dernière étape sur cette boucle est de définir le PID de T_c afin d'avoir en sortie une poussée respectant nos limitations et surtout un temps de réponse très court pour une réactivité maximale.

PID grande boucle

Intéressons nous désormais à la configuration des deux PD de la grande boucle concernant les erreurs sur x et y. Nous les configurons avec la même méthode que ceux de la petite boucle. Afin de respecter la symétrie du drone nous donnerons les mêmes valeurs pour les deux, ainsi qu'une saturation avec un minimum de -0,5 et un maximum de 0,5.

Enfin, nous ajoutons un bloc de saturation qui va venir limiter la rotation des moteurs en sortie, à la fin du sous-système Simulink "Représentation de la poussée" et ainsi protéger le drone de sollicitations mécaniques trop importantes susceptibles de nuire à sa durabilité.

Résultat sur un déplacement quelconque

La configuration de nos PID ne vise pas à être parfaitement optimale, dans le sens d'un équilibre entre rapidité et stabilité. Nous faisons délibérément le choix maximiser la réactivité pour correspondre aux possibilités de notre drone d'un pilotage dynamique et de suivi de trajectoire agressive. Mais de par ce choix, nous négligeons, ou du moins nous minimisons l'amortissement, l'effort de contrôle, la consommation d'énergie et la stabilité. De parce choix, nous avons des réactions très rapides, de mouvements impulsifs.

Mais nous avons aussi des conséquences, nous ne perdons pas complètement la stabilité du drone mais la marge de stabilité est proche de sa valeur max, les commandes fortes usent les moteurs et consomme plus d'énergie, ce qui réduit l'autonomie et de légères oscillations et dépassements surviennent après un virage. Mais au vu des performances que nous devons atteindre, nous pensons que cela reste un bon compromis.

Voici ainsi le tableau des gains PID pour chaque axe et angle, obtenus après de nombreux tests et trouver le meilleur compromis possible pour notre cas.

TABLE 1 – Valeurs des gains PID pour chaque axe

Axe	k_p	k_i	k_d	N
y	0.0313	–	0.1126	20.7401
x	0.0313	–	0.1126	20.7401
z	0.9722	–	1.9684	69.8475
ϕ	0.6089	0.0114	0.6058	15.1601
θ	0.6089	0.0113	0.6058	15.1601
ψ	0.554 46	0.015 74	0.421 77	12.732 25

En effectuant une simulation d'un déplacement quelconque du drone nous obtenons ces 4 graphiques en sortie des commandes.

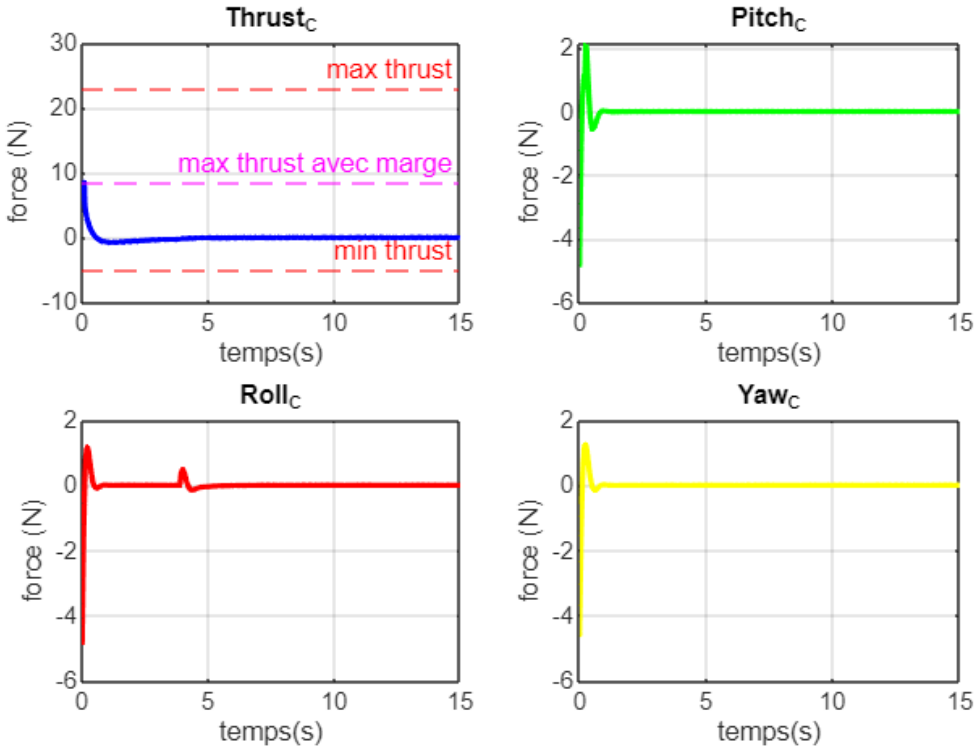

FIGURE 6 – Commandes moteur en newton

Bien que notre drone présente des caractéristiques favorisant une forte réactivité, nous constatons comme dit plus tôt que

sa marge de stabilité reste proche de la valeur maximale.

Concernant les temps de réponses des 4 graphiques, nous constatons une nette diminution de ceux-ci par rapport à un drone plus équilibré réglé avec des PID strictement optimaux (comme le PHANTOM 4 PRO V2) en étant deux fois plus rapide. Cette performance est cohérente avec le commentaire trouvé sur la fiche technique du drone PARROT expliquant qu'il est deux fois plus réactif que des drones deux fois lourd que lui. Nous nous doutons que cette indication est à but commercial mais elle démontre une certaine confiance dans les performances que nous avons voulu retranscrire dans ce projet.

Comparaison avec les résultats du TP2 après configuration des PID

Après avoir configurer nos PID nous pouvons effectuer une analogie avec le TP2 et évaluer leur impact sur le système.

Nous effectuons ici un déplacement d'un point A de coordonnées :

$$P_A = [0,\ 0,\ z_{\text{ground}} + 10,\ 0]$$

à un point B de coordonnées :

$$P_B = \left[120,\ -9,\ z_{\text{ground}} + 8,\ \frac{\pi}{4}\right]$$

puis nous affichons l'évolution de nos variables durant le déplacement.

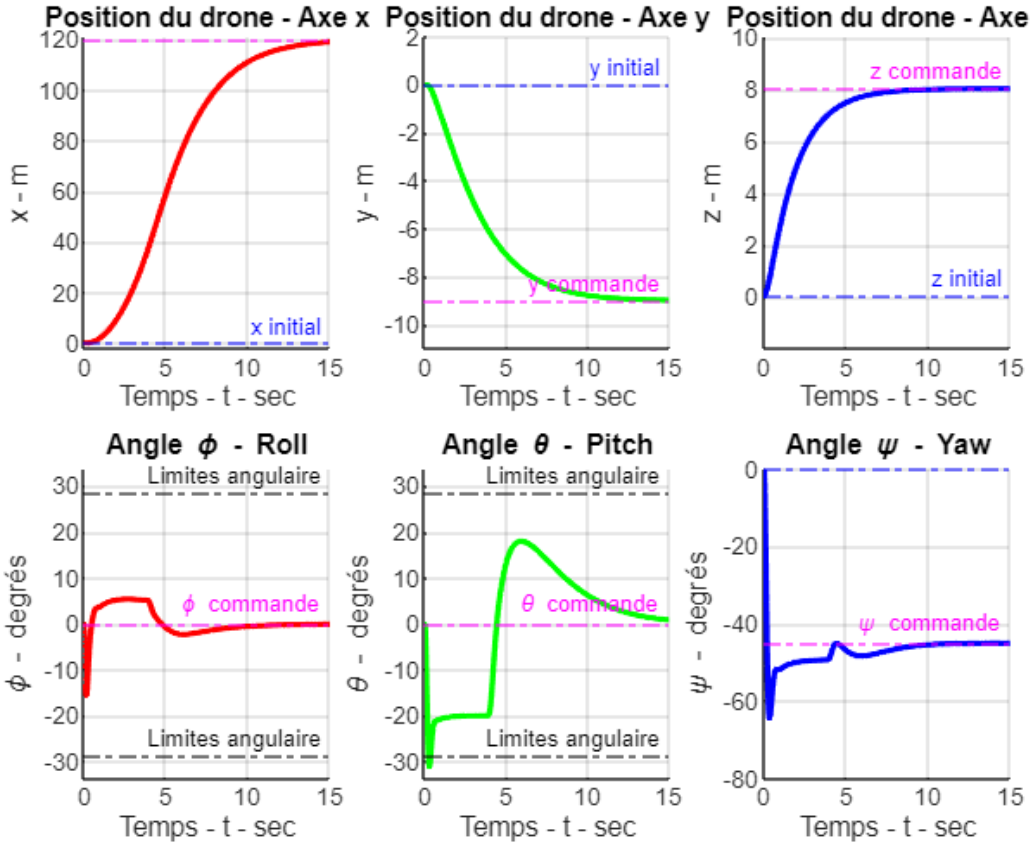

FIGURE 7 – Evolution des variables durant le déplacement

Concernant la position (x,y,z) nous n'observons plus de dépassement des valeurs cibles, au contraire les courbes viennent épouser celles-ci résultant ainsi d'un mouvement plus fluide et réaliste.

Les angles (θ, ϕ, ψ) se voient naturellement modifiés par l'ajout de ces PID. Nous remarquons des oscillations rapides de nos angles témoignant de la réactivité du drone. De plus le caractère rectangulaire du TP2 est beaucoup moins présent mettant en évidence la diminution du comportement on/off du système rejoignant l'idée de fluidité et de stabilité apporté par les PID. De plus, à la fin du déplacement nous retournons à une position en vol stationnaire stable ce qui montre le caractère fonctionnel de notre simulation de drone.

TP4 : Suivi d'une trajectoire

Cette dernière phase vise à mettre en application l'ensemble des principes développés lors des précédentes phases à l'aide d'un scénario de navigation autonome. L'objectif est de concevoir une machine à états capable de piloter le drone tout au long de son parcours, en respectant un ensemble de règles précises.

Cela implique de guider le drone à travers l'ensemble des 26 points de passage (waypoints) imposés, tout en s'assurant qu'il contourne efficacement les obstacles rencontrés en vol. Il est également essentiel qu'il conserve une orientation stable tout au long du parcours, pour finalement achever sa mission par une arrivée précise à l'emplacement prévu.
Le système repose sur une architecture à états discrets, où chaque étape du vol correspond à un état spécifique déclenché par des conditions bien définies. Cette organisation permet de séquencer le comportement du drone de manière logique et maîtrisée, en visant une exécution autonome, fiable et fluide du parcours programmée ci-dessous :

FIGURE 8 – Trajet de *Race.mat*

Implémentation d'un contrôleur Stateflow pour le suivi de parcours

Pour permettre au drone de suivre une trajectoire définie, nous pouvons utiliser une modélisation par états mise en place sous Matlab à l'aide de l'outil Stateflow. Nous ne faisons pas un trajet complet et continue, cette approche permet de fixer des waypoints donnant des commandes que nous pouvons enregistrer dans un fichier, comme ici avec *Race.mat*. Chacune de ces commandes est ensuite déclenchée en fonction de la position actuelle du drone.

Les coordonnées des points à atteindre sont introduites en entrée du modèle, accompagnées de seuils spécifiques servant à valider leur franchissement. Grâce à ces données, le système évalue en continu la distance au prochain objectif et ajuste la direction du drone en calculant l'erreur associée au prochain waypoint pour parcourir tout le chemin sans rater de points.

Nous avons aussi un suivi temporel intégré permettant de relever le temps mis pour parcourir chaque segment. Ainsi, cette architecture nous permet de garder une prédiction précise de la trajectoire tout en pouvant être utilisé avec la structure que nous avons construite depuis le début.

Données utiles durant le parcours réalisé par le drone pendant la simulation

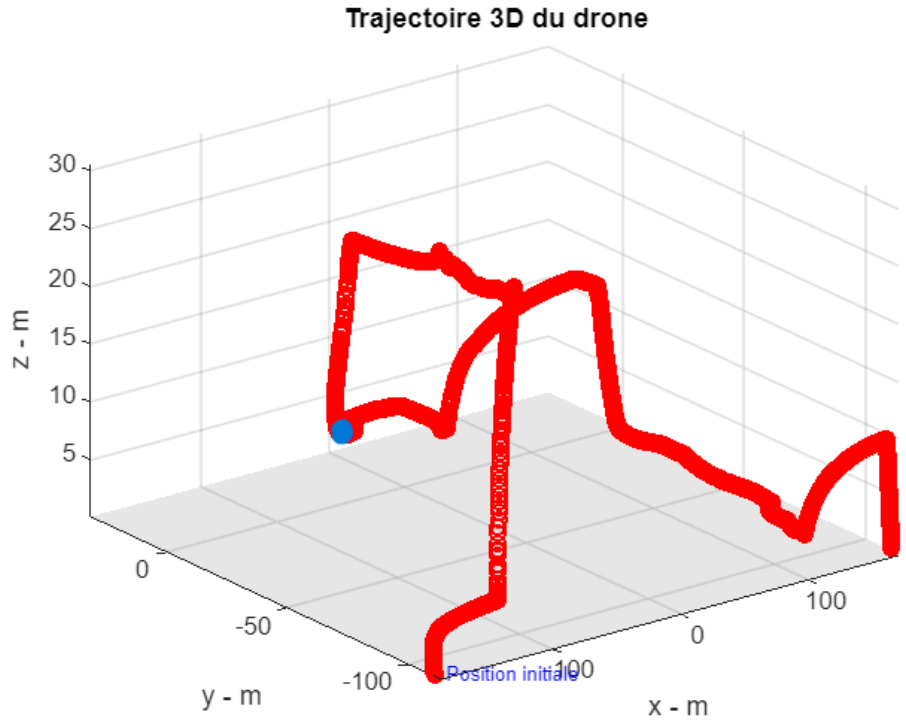

FIGURE 9 – Evolution de la position 3D durant le parcours

Durant notre simulation, nous remarquons que le drone a une petite erreur de précision où il va venir effleurer un poteau avec ses hélices du coté droit. Nous pouvons expliquer cela par la trajectoire prise par le drone qui est directement lié à sa réactivité. Cependant, même si elle reste gênante et qu'une amélioration est possible, cette légère collision ne remet pas en question l'opérationnalité du drone.

De plus, nous avons l'évolution de la distance entre le drone et le point suivant à atteindre :

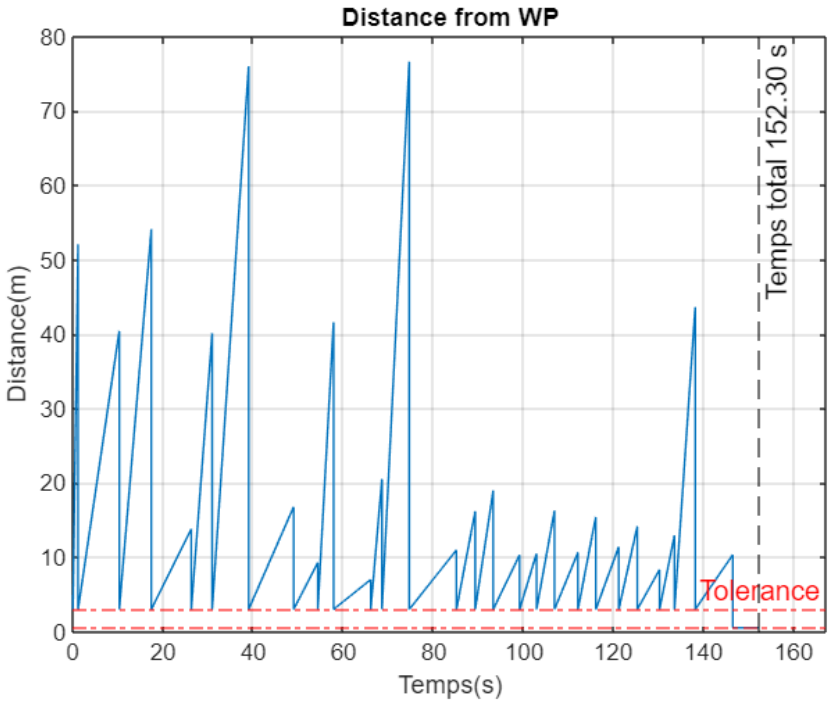

FIGURE 10 – Evolution de la distance entre le drone et le point suivant à atteindre

Nous observons vingt-cinq rebonds sur la ligne de tolérance de trois mètres. En effet, dès que le drone atteint une distance absolue de trois mètres, l'outil stateflow passe à l'itération suivante, indiquant ainsi au drone la prochaine position à atteindre. La simulation s'arrête automatiquement avec P_{finale} atteint avec une tolérance de 0,2 mètre, où le drone est censé atterrir.

De plus, nous constatons un temps de parcours de 152 sec, ce qui est significativement plus court comparé à la simulation du même parcours avec le drone PHANTOM 4 PRO V2 qui est de 230 sec bien que les PID n'aient pas forcément été optimisés au maximum sur celui-ci. Cela constitue une réussite pour notre équipe et notre objectif d'avoir un drone typé "race". Ce temps de parcours pourrait être bien plus court si la validation de chaque waypoint n'imposait pas un ralentissement et un passage en vol stationnaire du drone pendant une fraction de seconde en arrivant aux coordonnées. Nous pourrions donc passer de waypoint à des checkpoint où simplement passer dessus validerait l'étape.

Nous présentons également l'évolution du temps de passage sur chaque segment du parcours :

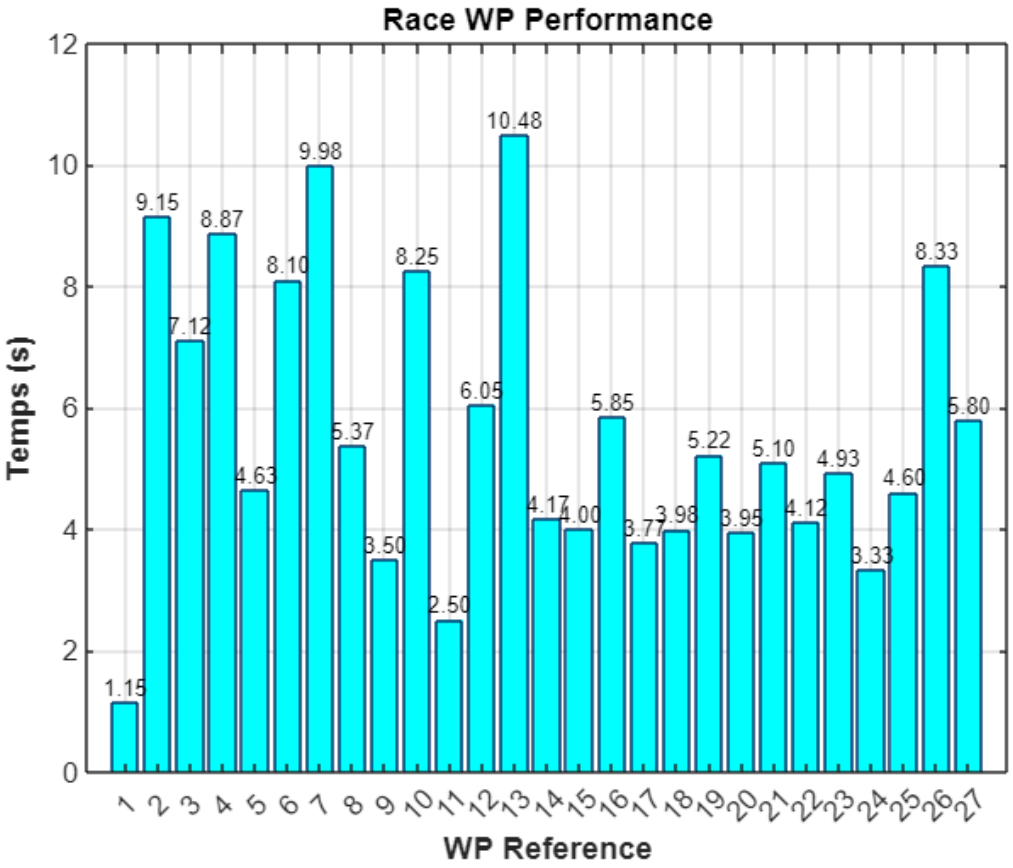

FIGURE 11 – Evolution du temps de passage sur chaque segment du parcours

Nous observons des variations importantes du temps de passage entre chaque waypoint, ils varient de 1,15 sec à 10,48 sec. Cela s'explique principalement par la distance séparant deux waypoint. Plus ceux-ci sont espacés dans l'espace plus le temps de vol sera long. Nous pouvons donc voir une analogie entre le graphique "Distance from Waypoint" et le "Race WP Performance". Cependant la distance séparant chaque waypoint n'est pas le seul facteur influant sur le temps de passage, en effet, la complexité entre chaque segment joue un rôle majeur. Par exemple un segment court en terme de distance mais comprenant un virage serré par rapport au segment précédent contraindra le drone à ajuster son orientation de manière plus agressive tout en restant stable ce qui augmentera donc son temps de passage sur celui-ci. Nous observons ce phénomène durant le passage du parc.

Conclusion

Au cours de ce projet, nous avons modélisé un drone quadrirotor (Parrot Anafi USA) en utilisant les outils MATLAB et Simulink. Cette expérience nous a permis de comprendre le fonctionnement du drone, sa stabilisation, son déplacement et les améliorations possibles. Chaque étape a présenté des défis spécifiques, mais celle qui nous a le plus gêné a été l' optimisation des différents PID du TP3, encore plus en prenant en compte les performances du drone que nous avons simulé. Cela était notre choix de ne pas prendre le même modèle de drone que le sujet, et nous avons apprécié pouvoir travailler librement sur le Parrot, mais devoir retrouver et adapter chaque paramètre fourni nous a pris un certain temps. Mais nous avons du faire face à d'autres difficultés, comme comprendre la représentation du drone avec les matrices dans le TP1 ou les signes à appliquer à chaque angle dans le TP2. Les changements d'ordres des angles ne nous aidaient pas à bien comprendre où se trouvait le problème dans notre modélisation.

En ce qui concerne les résultats, nous avons réussi à obtenir un drone très réactif tout en restant stable et cohérents avec les vitesses max du drone modélisé (cf. annexe 2) capable de suivre une trajectoire définie évitant presque toutes les collisions avec des obstacles. La seule collision que nous observons est un léger effleurement avec un lampadaire au cours de la simulation, c'était un passage qui nous avait été mentionné par le client, expliquant à quel point il pouvait être serré. Pour améliorer les performances, nous avons privilégié la vitesse, acceptant des dépassements mineurs sur les commandes angulaires. Malgré notre choix, pour aller plus loin il serait intéressant de trouver une solution à cela en trouvant peut-être des valeurs de PID encore plus axé sur la réactivité, avec un ordinateur plus puissant pour ne pas être limité comme cela a pu être le cas dans nos simulations quand nous mettions des valeurs trop grandes. D'après nos recherches, nous pourrions aussi nous tourner vers une autre méthode de modélisation ce qui pourrait faciliter le bureau d'étude afin d'obtenir un drone toujours aussi réactif mais ne mettant pas ses contraintes aux limites pour faire face à différentes condition de la vie réelle (vent, charge, choc...).

Nous avons aussi eu pour idée d'essayer de supprimer l'arrêt du drone à chaque waypoint que nous mentionnons dans le TP4. En plus de faire perdre beaucoup de temps sur le trajet global, cela ne ressemble pas à une parcours de course quand nous regardons la simulation. Mais la priorité était de finir dans le temps le projet demandé, nous avons donc dû écarter cette amélioration et la laisser pour une possible ouverture.

Pour rendre la modélisation plus réaliste, plusieurs améliorations pourraient être envisagées. Par exemple, prendre en compte la traînée et les frottements de l'air dans les équations du mouvement, ou encore intégrer des compensateurs pour pallier les effets de l'environnement. Il serait également intéressant d'avoir un modèle 3D du Parrot Anafi USA plutôt que le modèle utilisé durant la simulation qui a plus le gabarit d'un DJI de la gamme Phantom.

En définitive, ce projet nous a permis de mieux appréhender les enjeux de la modélisation et du contrôle d'un drone. Il offre une très bonne base pour envisager des développements plus poussés. Il pourrait être intéressant d'utiliser le drone dans la réalité en lui faisant faire un parcours similaire et enregistrer son comportement. Cela nous permettrais d'avoir plus de données pour estimer si les résultats obtenus sont en adéquations avec ce que nous avons.

Annexes

Annexe 1 : Simulink au terme du TP4

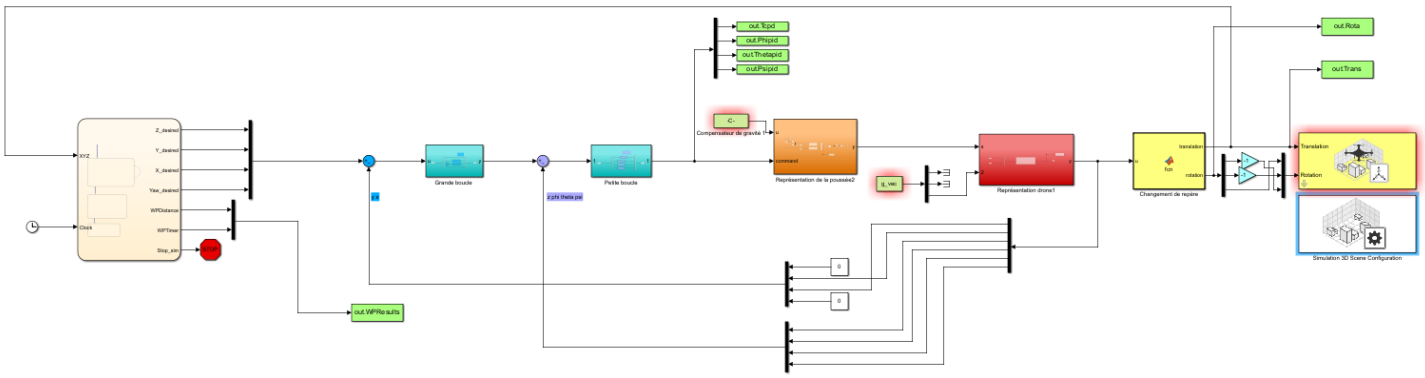

FIGURE 12 – Simulink Global

L'architecture générale de notre modèle Simulink est organisés en sous blocs chacun ayant son propre rôle bien défini pour le bon fonctionnement de la simulation.

- **Bloc beige** : Ce bloc est le point d'entrée du système. Il permet de récupérer les checkpoints à atteindre par le drone, grâce à sa position en entrée.

- **Blocs bleus** : Ces blocs représentent deux boucles corrections :
  - La grande boucle effectue le calcul de l'erreur de position en x et y, corrigée grâce à deux régulateurs PID, pour créer une commande en orientation.
  - La petite boucle corrige les angles d'attitude (ϕ, θ, ψ) à l'aide de quatre PIDs supplémentaires.

- **Bloc orange** : Ce bloc reçoit en entrée la commande issue des boucles précédentes ainsi qu'un compensateur de gravité. Son rôle est de générer les commandes moteurs en sortie

- **Bloc rouge** : Il représente le drone. Ce bloc reçoit les commandes moteur issues du bloc orange, ainsi que le vecteur gravité $\vec{g}$. Il renvoie la *représentation d'état* du drone.

- **Blocs jaunes** : Ils assurent les changements de repère et la simulation du drone. Ces blocs permettent d'obtenir l'aperçu final.

- **Blocs verts** : Ce sont les sorties du système. Elles sont utilisées pour obtenir des données de calcul et tracer nos graphiques. Cela nous permet d'analyser le comportement du drone.

Annexe 2 : Fiche technique drone

ANAFI USA

White Paper

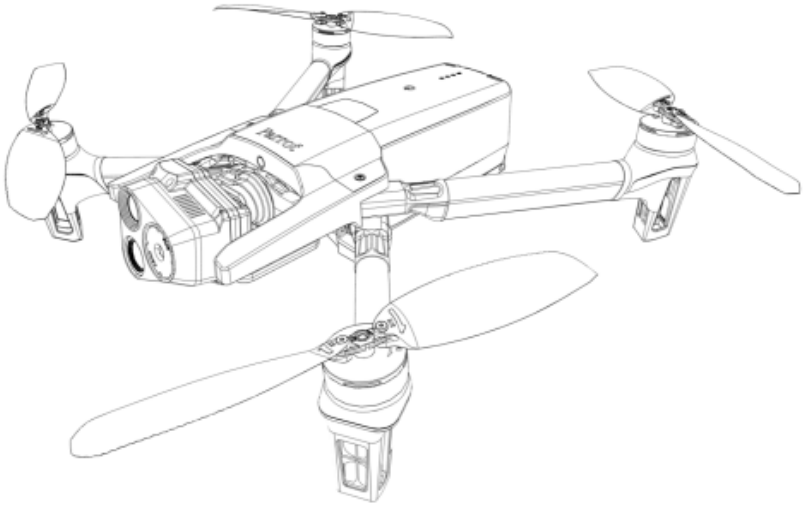

Figure 13 – Représentation du parrot Anafi USA

Caractéristiques :

- Dimensions plié : 252 x 104 x 84 mm
- Dimensions déplié : 282 x 373 x 82 mm
- Dimensions déplié (version MIL) : 282 x 373 x 192 mm
- Masse : 500 g / 1,10 lb
- Masse maximale au décollage (MTOM) : 644 g / 1,42 lb
- Autonomie maximale : 32 minutes (30 minutes pour la version MIL)
- Vitesse horizontale maximale : 14,7 m/s
- Vitesse de montée maximale : 4 m/s (6 m/s pour versions SE, GOV, MIL)
- Vitesse de descente maximale : 3 m/s
- Résistance au vent maximale : 14,7 m/s
- Vitesse maximale des hélices : 11 000 tr/min
- Niveau sonore à 1 m : 84 dB
- Plafond de service : 5 000 m au-dessus du niveau moyen de la mer (MSL)
- Géo-clôturage défini par l'utilisateur
- Certifié IP53 : résistant à la poussière et à la pluie
- Température de fonctionnement : de -36 °C à +50 °C
- Aucune limite de température pour le décollage