RAPPORT FINAL - PSC BACKONEARTH - MEC012

2022

Julie Albigot, Thomas Barreira, Guillaume Faye-Bédrin,
Virgile Foussereau, Antoine Fropo

Remerciements

Nous souhaitons commencer ce rapport en remerciant toutes les personnes qui nous ont aidés au cours de ce projet, tant par leurs expertises techniques que relationnelles.

Nous remercions tout d'abord Laurence Bodelot, notre coordinatrice, de nous avoir encadrés durant toute cette année, d'avoir été notre référente auprès de l'École Polytechnique et pour ses commentaires constructifs concernant les rapports.

Nous poursuivons en remerciant Ruben Di Battista qui, en tant que tuteur, nous a apporté une aide technique et un soutien moral durant tout ce projet. Nous le remercions également de sa rapidité à valider nos demandes de financement ou de commandes de matériel.

Nous adressons également de chaleureux remerciements À Cyril Hasson et Gareth Paterson qui nous ont formés et permis d'accéder aux installations et machines de l'espace de prototypage du Drahi'X pour réaliser notre projet. Gareth Paterson a été particulièrement présent pour nous dès qu'il s'agissait de CAO et d'impression 3D. Cyril Hasson, quant à lui, nous a apporté une aide précieuse pour le design et lors de la construction de notre second banc de tests.

Poursuivons en remerciant Antoine Tavant et l'ensemble des membres du Centre Spatial Étudiant de Polytechnique pour les moyens humains et logistiques qu'ils ont mis à notre disposition ainsi que William Gilbert du laboratoire de mécanique pour avoir accepté de nous prêter un dynamomètre, essentiel aux expériences de notre projet.

Enfin, nous remercions l'École Polytechnique d'avoir mis à notre disposition des moyens pour avancer dans notre projet. Nous pensons également à l'ensemble du personnel administratif qui nous a encadrés et particulièrement à Sylvie Pottier, gestionnaire de l'école, d'avoir géré nos commandes externes.

Nous aimerions enfin adresser un dernier mot aux élèves polytechniciens des promotions antérieures qui ont travaillé sur ce projet et à qui nous souhaitons dire notre fierté d'avoir poursuivi le travail.

Sommaire

Introduction

Le PSC *Back on Earth* sur lequel notre groupe a travaillé est né en 2019 sous l'impulsion d'une équipe menée par Arnaud Ballande, de la promotion X2018. Leur souhait était de concevoir un prototype de fusée électrique réutilisable. Le projet a été repris par une équipe de X2019, pour qui la situation sanitaire a compliqué la tâche, puis par notre groupe.

Ce projet s'inscrit dans l'air du temps : l'industrie spatiale a été bouleversée par l'arrivée des lanceurs réutilisables conçus par SpaceX, avec l'appui de la NASA, dans les années 2010. Les promesses du réutilisable sont multiples : coûts réduits et donc accès facilité à l'espace, moins de perte-sèche à chaque lancement et moins de gaspillage, sécurité accrue sur des lanceurs ayant déjà démontré leur fiabilité.

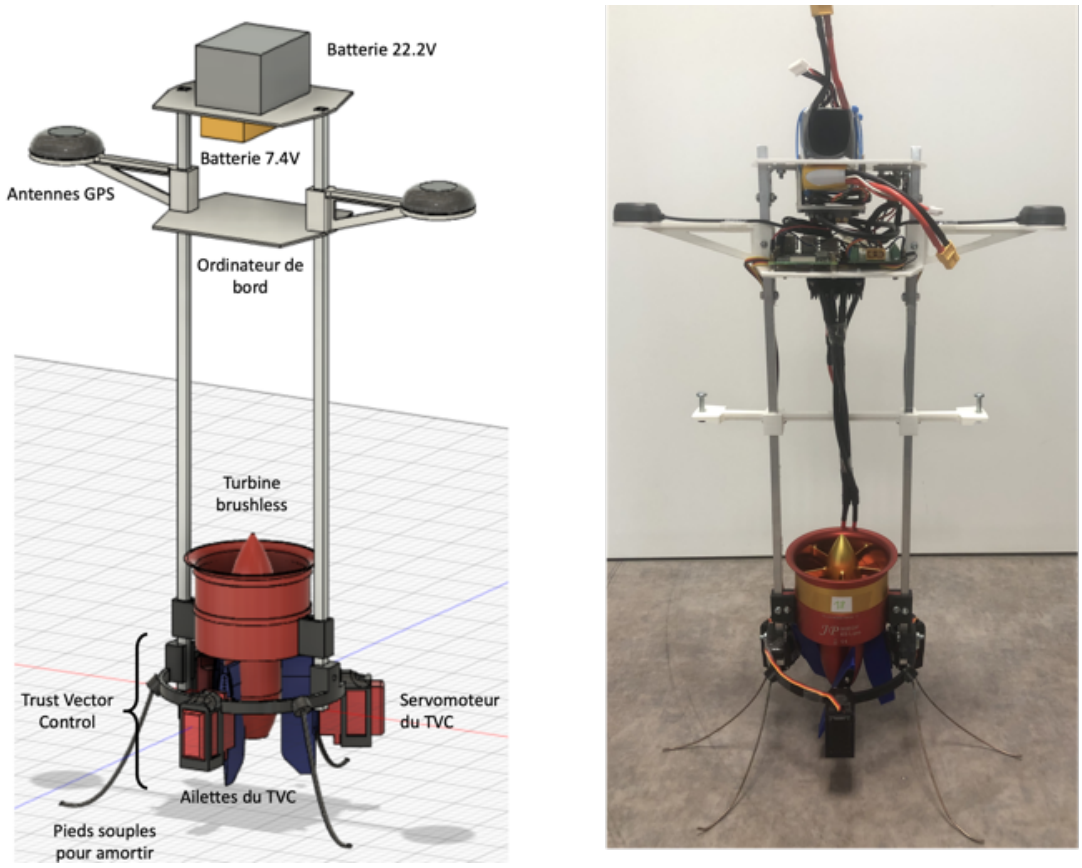
La NASA a notamment revu sa stratégie en s'appuyant de plus en plus sur les lanceurs réutilisables de SpaceX au détriment de son propre lanceur Space Launch System (SLS, fusée lourde non réutilisable) qui tarde à entrer en service et qui est critiquée pour son coût de développement très élevé. C'est notamment une fusée réutilisable développée par SpaceX, le Starship, qui sera chargée d'amener les astronautes du programme Artemis sur la surface lunaire. L'Europe a fait preuve de prudence face à cette technologie dont les promesses économiques restaient à prouver. Cette dynamique a cependant changé ces dernières années. Fin 2020, l'Agence spatiale européenne (ESA) signe un contrat avec ArianeGroup ayant pour objet le développement d'un démonstrateur de lanceur réutilisable, Thémis. Fin 2021, le ministre de l'Économie et des Finances fait l'annonce du développement par ArianeGroup d'un nouveau lanceur léger réutilisable, nommé Maïa, pour l'horizon 2026.

La capacité à contrôler et faire ré-atterrir une fusée est ainsi une technologie de pointe plus que jamais d'actualité, ce qui met en exergue la pertinence du projet *Back on Earth* au sein de l'École polytechnique. Notre démarche s'articule donc naturellement en plusieurs étapes. Dans un premier temps, nous nous somme familiarisés avec le projet et avec le matériel que nous ont laissé l'équipe X19. L'un de leurs membres, Alexandre Kirchmayer, nous a beaucoup aidés dans cette tâche. Ceci fait, nous avons repris leurs travaux là où ils les avaient laissés. Le premier objectif était d'achever la conception et l'assemblage de la fusée. Cet objectif a été rapidement rempli. Les détails du fonctionnement sont décrits dans la partie 1. La seconde étape était de valider l'architecture de la fusée en réalisant un premier vol captif, et de régler le *Thurst Vectoring Control* (TVC), c'est-à-dire le système d'ailettes permettant de diriger la fusée, et de déterminer une loi entre l'orientation des ailettes et le couple engendré sur la fusée. Cet objectif est l'objet de la partie 2. Enfin, dans l'objectif de faire voler la fusée, nous avons dû dans un dernier temps la doter d'un algorithme de contrôle, détaillé dans la partie 3.

1 Fonctionnement de la fusée

1.1 Présentation générale

L'architecture mécanique et électronique de la fusée est composée de différents modules. Ceux-ci sont légendés sur le modèle 3D de la fusée en figure 1a.



(a) Modèle 3D de la fusée

(b) Photo de l'architecture de la fusée

Figure 1: Structure actuelle de la fusée

En partant du haut de la fusée, on peut observer figure 1 les deux batteries, l'une en 22,2 V pour alimenter le moteur principal et l'autre en 7,4 V pour fournir l'énergie nécessaire à l'ordinateur de bord et aux servomoteurs.

Sous les batteries se trouve le cerveau de la fusée, constitué d'une carte Raspberry Pi comme ordinateur de bord, ainsi qu'un contrôleur électronique de vitesse ou *Electronic Speed Controler* (ESC) qui régule la puissance à fournir au moteur principal, et un circuit imprimé ou *Printed Circuit Board* (PCB) qui sert à aiguiller les informations de commande sortant de la Raspberry vers les différents récepteurs électriques. Nous détaillerons son fonctionnement dans la partie

dédiée à l'électronique de la fusée, section 1.3.

À ce niveau se situe également une centrale inertielle, ou *Inertial Measurement Unit* (IMU), reliée à ses deux antennes de part et d'autre la fusée. Cette centrale inertielle a été fournie au groupe de la promotion X2019 dans le cadre d'un partenariat avec la firme VectorNav. Son fonctionnement est détaillé section 1.4.

On observe au milieu de la fusée une pièce blanche, présente sur la photographie (figure 1b) mais absente sur le modèle 3D (figure 1a). Cette pièce, dont le modèle 3D est montré figure 11, ne fait pas partie de la fusée à proprement parler mais sert à fixer celle-ci sur le banc de test que nous avons conçu au cours de l'année et qui est présenté section 2.3.

Dans la partie inférieure de la fusée se trouve le module de propulsion et d'orientation de la fusée. Il est constitué en premier lieu du moteur principal qui actionne une hélice enchâssée dans un carénage. Celui-ci, même s'il augmente le poids total de la fusée, améliore significativement les performances du moteur, comme l'a mis en évidence le groupe de la promotion X2019.

Sous le moteur, c'est-à-dire à la sortie du flux d'air, se trouve le module d'orientation de la fusée, que nous appelons *Thurst Vectoring Control* (TVC). Il est composé de trois servomoteurs auxquels sont attachés des ailettes imprimées en 3D qui permettent l'orientation du flux d'air donc la manœuvrabilité de la fusée.

Enfin, trois pieds formés de tiges métalliques flexibles, ce qui leur permet d'être souples, légers et résistants, permettent à la fusée de conserver sa position verticale lorsqu'elle est posée sur le sol, et d'amortir les atterrissages.

La figure 1b permettra au lecteur de se faire une idée de l'agencement de ces différents modules sur le prototype de la fusée. Ainsi assemblée, la fusée pèse 2 kg. Pour un engin volant tel que notre prototype, la légèreté est un atout capital, et c'est pourquoi une partie de notre travail durant cette année a consisté à alléger la fusée des pièces superflues, ce qui est l'objet de la section 1.6.1.

1.2 Alimentation de la fusée

Le bon fonctionnement de la fusée nécessite deux batteries lithium-ion polymère (Li-Po). La première, une batterie de 6 éléments en 22,2 V pour 1,55 Ah, fournit la haute puissance pour le moteur principal. La seconde, constituée de deux éléments en 7,4 V pour 2,2 Ah, alimente les récepteurs de faible puissance, c'est-à-dire l'ordinateur de bord, l'IMU et les servomoteurs.

Concernant l'autonomie, la capacité limitante est celle de la batterie 6S qui alimente le moteur. Avec ses 1,55 Ah, celle-ci permet une utilisation de la fusée en conditions de vol (poussée moteur d'au moins 60 à 70 %) durant une minute à une minute et demie, comme l'a montré

l'expérience décrite section 2.1.1. Comme nous possédons 2 batteries de ce type, nous disposons au maximum de 3 minutes d'essais ou de vol avant de devoir recharger les batteries.

Pour augmenter la durée de nos essais au sol utilisant le moteur en conditions de vol, nous avons pensé à alimenter ce dernier via une alimentation externe branchée sur le secteur. Malheureusement, nous avons abandonné cette idée et nous en expliquons les raisons dans le paragraphe sur la motorisation 1.5. À ce jour, l'unique solution est d'utiliser de grosses batteries 22,2 V de 5 Ah. Celles-ci étant trop lourdes pour permettre le vol, nous les réservons pour les essais au sol, durant lesquels nous plaçons une batterie de 1,55 Ah à son emplacement prévu pour ne pas modifier la géométrie et la répartition de masse de la fusée durant les tests. Durant de telles expérimentations au sol, toutes les batteries 22,2 V peuvent être utilisées pour alimenter le moteur ce qui permet d'atteindre une autonomie d'environ 15 minutes.

1.3 Électronique de commande de la fusée

Le cœur de commande de la fusée est la Raspberry Pi. Il s'agit d'un petit ordinateur fonctionnant sous Linux, et contrôlable via une interface graphique en HDMI ou via une connection WiFi en *SecureShell* (SSH). L'utilisateur transmet à la Raspberry des commandes pour exécuter les programmes situés dans sa mémoire interne. La Raspberry fournit en sortie des signaux de commande numériques ou analogiques. Ceci permet de brancher nos récepteurs (ESC et servomoteurs) aux sorties de la Raspberry pour pouvoir les commander. Ceux-ci utilisent des signaux à modulation de largeur d'impulsion ou *Pulse Width Modulation* (PWM) en consigne, mais ce signal est interprété différemment par les servomoteurs et par le moteur.

Le signal PWM est un signal créneau dont la fréquence et la largeur des créneaux sont modulées pour obtenir un signal pseudo-analogique. Un servomoteur fonctionne en contrôle de la position, et c'est la largeur des créneaux qui commande la position. La fréquence peut varier (de 40 à 200 Hz) sans que la position du servomoteur ne varie pour autant. Comme cela est expliqué dans le document [1], la largeur des créneaux va de 500 µs (position minimale) à 2500 µs (position maximale). La position angulaire suit une loi affine en la largeur des créneaux. Au contraire, le moteur dédié à la propulsion fonctionne en contrôle de la vitesse, donc la fréquence du signal PWM détermine la vitesse de rotation. Plus précisément, la vitesse de rotation est proportionnelle à la valeur moyenne du signal, donc au rapport des largeurs des créneaux et des creux.

Détaillons la méthode choisie pour communiquer avec la Raspberry. Lorsque nous avons hérité du projet des X2019, la Raspberry était paramétrée pour se connecter automatiquement au réseau *Eduroam*. Nous pouvions donc, depuis un ordinateur connecté à ce même réseau, nous connecter par le protocole SSH à cette dernière afin de modifier et implémenter des programmes et envoyer des ordres aux différentes sorties de la Raspberry. Cependant, la Raspberry avait tendance à se déconnecter du réseau WiFi et à changer d'adressse IP de manière intempestive, ce qui ne permettait pas un contrôle fiable de la fusée. Nous avons donc reparamétré

la Raspberry pour qu'elle se connecte à un téléphone, dont l'adresse IP est fixe, en partage de connexion. Depuis ce reparamétrage, la liaison entre l'ordinateur de contrôle et la Raspberry est beaucoup plus fiable.

Nous avons réalisé un calibrage de la position neutre des servomoteurs telle que ceux-ci n'exercent aucune force latérale sur la fusée sous l'effet du flux d'air. Comme nous l'attendions, cette position neutre correspond, pour la largeur des créneaux du signal PWM, à une valeur proche de 1500 µs, valeur moyenne entre 500 et 2500 µs. Nous avons en effet constaté que l'écart à cette valeur moyenne est inférieur à 50 µs. Nous mesurerons l'effet d'une commande de position des servo-moteurs grâce au banc de tests que nous avons confectionné et dont la structure est détaillée en partie 2.3.

Enfin, pour rendre opérationnel le contrôle de la fusée, nous avons achevé le branchement et l'agencement de l'ensemble des modules électroniques, commencés par l'équipe précédente, selon le schéma électrique présenté en annexe A.

1.4 Système de navigation de la fusée : l'IMU

La centrale inertielle, ou *Inertial Measurement Unit* (IMU), est le capteur électronique donnant accès à la position et l'attitude de la fusée pour permettre ensuite à l'algorithme de contrôle de corriger la trajectoire. Le fonctionnement de ce composant repose sur deux sous-systèmes [2] [3].

D'une part, grâce au système de navigation inertielle, ou *Inertial Navigation System* (INS), il est capable de mesurer l'accélération à laquelle il est soumis, et de calculer par intégration la vitesse et position de l'engin. Sur la fenêtre *Sensor Status* de la figure 2, on trouve un exemple d'estimation de vitesse et de position par l'INS, ainsi que de leurs incertitudes respectives. Ce sytème fonctionnant par calcul intégral, l'erreur s'accumule au cours du temps.

Afin de corriger ce défaut, l'IMU comporte un système de navigation par satellite, ou GNSS *Global Navigation Satellite* (GNSS). Lorsque la visibilité du ciel est bonne, la position obtenue grâce à ce système est assez précise pour corriger en permanence la position fournie par l'INS [4]. Ce dernier reste cependant suffisamment fiable en cas de perte de signal satellite. Ainsi, les deux systèmes se complètent pour obtenir des valeurs de position et de vitesse précises et fiables au cours du temps.

Nous avons pris en main la bibliothèque Python développée par VectorNav pour l'IMU afin d'extraire les données nécessaires au calcul de la trajectoire par les algorithmes (angles et vitesses dans les trois directions, altitude et vitesse verticale). L'IMU est donc à présent fonctionnel et au service de l'ordinateur de bord.

Figure 2: Interface graphique de paramétrage de l'IMU

Comme expliqué en section 1.6.1, afin d'alléger la fusée, nous avons commandé de nouvelles antennes GPS pour l'IMU. Suite à la réception de celles-ci, nous avons indiqué leur position relative dans le référentiel de l'IMU.

Nous avons également changé l'orientation du référentiel de référence dans la mémoire de l'IMU pour l'adapter au référentiel choisi pour la fusée. Ces paramètres sont visibles sur l'extrait de l'interface VectorNav de la figure 2. Il est à préciser que sur la fenêtre de droite montrant une vue 3D, les flèches rouge, verte et bleue définissent les axes par défaut de l'IMU et non ceux choisis pour le référentiel de la fusée. L'orientation de ceux-ci n'est pas conforme à celle que nous avons choisi, et c'est pourquoi nous avons décidé de changer le référentiel de référence dans la mémoire interne de l'IMU.

1.5 Motorisation

La fusée a été conçue pour être propulsée verticalement par une turbine électrique. La propulsion électrique possède en effet certains avantages sur la propulsion à ergols classiques : la masse de la fusée ne varie pas au cours du vol, la propulsion électrique est plus respectueuse de l'environnement et surtout la puissance d'une turbine est beaucoup plus facilement réglable que la puissance d'une tuyère à ergol. L'objectif de ce PSC étant la conception d'un algorithme de contrôle, et non pas la conception d'une maquette réaliste de lanceur orbital, une

telle propulsion est donc pertinente.

Nous avons décidé de conserver la motorisation que nous avons héritée du projet du groupe de la promotion X2019. Ceux-ci ont beaucoup travaillé sur cet aspect, car ils ont initialement fait face à un déficit de poussée considérable par rapport à la valeur attendue. Grâce à leur travail, nous avons hérité d'une fusée dont la nouvelle turbine avait une poussée en accord avec la valeur attendue et suffisante pour permettre à la fusée de décoller. C'est ce que nous avons vérifié lors de notre première expérience décrite section 2.1.1.

La turbine, illustrée figure 3, est composée d'un moteur brushless alimenté en 22,2 V triphasé sur lequel est branchée une hélice de 12 pales. Un carénage autour du moteur et un cône en entrée de tuyère permettent de canaliser le flux d'air de façon optimale, comme le montre la figure 3. La poussée nominale maximale de la turbine est de 3,5 kg. La vitesse de rotation de la turbine est proportionnelle à la fréquence du signal PWM reçu par l'ESC, voir section 1.3.

Figure 3: Turbine JP Hobby 12 pales

À cause des très courtes autonomies des batteries, de l'ordre de la minute comme vu en parties 1.2 et 2.1.1, nous avons cherché à alimenter le moteur via une alimentation externe branchée sur secteur pour fournir l'énergie nécessaire pendant une durée de test beaucoup plus longue. Nous avons pour cela mesuré l'intensité du courant traversant le moteur pour différentes vitesses de rotation afin de se faire une idée de la puissance de l'alimentation nécessaire. À 10 % de puissance moteur, l'intensité par phase est de 30 A. Celle-ci monte jusqu'à 80 A autour de 60 à 80 % de puissance moteur. Lors du pic de consommation énergétique due au démarrage du moteur, l'intensité dépasse les 100 A par phase. Ces fortes intensités nous ont fait abandonner l'idée d'une alimentation externe car celles que nous avons trouvées dans le commerce spécialisé ne fournissent au maximum que 30 A en 22 V.

1.6 Évolution de la structure de la fusée

1.6.1 Travail d'allègement de la fusée

Dès la reprise du projet par notre groupe, nous avons cherché à alléger la fusée des masses non essentielles. En effet, plus la fusée est légère, moins la puissance demandée au moteur pour décoller est importante et plus l'autonomie de la batterie est grande.

Nous avons tout d'abord gagné quelques dizaines de grammes en raccourcissant l'ensemble des câbles électriques dont les longueurs avaient été prévues beaucoup trop longues, notamment les câbles reliant les servomoteurs à la PCB, celui reliant l'IMU à la Raspberry et enfin les câbles liant les deux antennes GPS à l'IMU.

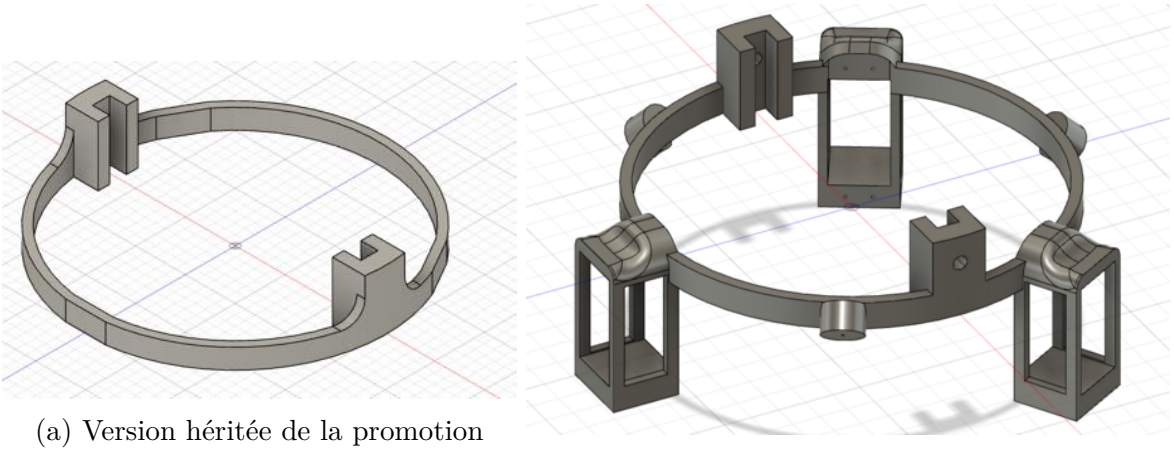
Nous avons également allégé la fusée de 340 g en changeant les antennes GPS de l'IMU. En effet, celles d'origine possédaient un lourd aimant interne permettant de les fixer à la carrosserie métallique d'une voiture par exemple. Elles pesaient à elles deux plus de 400 g. Nous avons donc commandé de nouvelles antennes chez Naelcom. Le poids cumulé de ces deux nouvelles antennes est d'environ 70 g.

Cet allègement de la fusée était indispensable pour pouvoir compenser l'ajout de nouveaux modules comme l'IMU, qui n'avait jamais été monté sur la fusée, ainsi que l'ajout de pieds et le renforcement de la pièce support du TVC, ce qui est l'objet de la section 1.6.2.

1.6.2 Renforcement du support TVC et ajout de pieds

Nous avons constaté plusieurs améliorations et modifications à effectuer sur la structure de la fusée que nous avons héritée du PSC de nos camarades de la promotion précédente.

Premièrement, nous avons modifié la pièce supportant les servomoteurs du TVC et joignant les deux tiges principales de la fusée. L'objectif était de la renforcer et de permettre l'ajout de pieds. Cette pièce se situe sous le moteur, au bas de la fusée, comme présenté en partie 1.1. Nous avons effectué trois modifications sur cette pièce, comme le montre la figure 4.



(a) Version héritée de la promotion précédente

(b) Version actuelle

Figure 4: Pièce de support du TVC

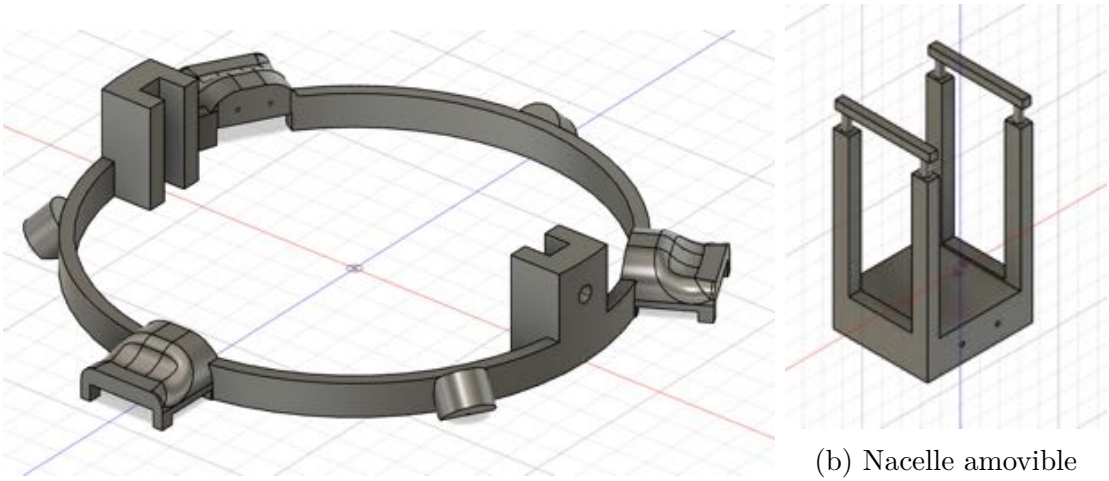
Nous avons d'abord ajouté trois supports permettant d'enchâsser des tiges métalliques flexibles pour jouer le rôle de pieds en prévision des décollages et atterrissages de la fusée.

Ensuite, nous avons ajouté des trous dans la partie de la pièce accueillant les montants verticaux métalliques. Cela permet de positionner les vis de manière précise, en réduisant les sources d'incertitude dues aux mesures manuelles servant à positionner les perçages.

Enfin, nous avons constaté que les servomoteurs du TVC portant les ailettes n'étaient pas fixés de manière satisfaisante : il y avait du jeu qui causait une orientation imprécise des ailettes et une instabilité. Pour régler ce problème, nous avons, dans un premier temps, significativement augmenté la surface de contact entre le support et les servomoteurs. Nous avons alors constaté que le gain en rigidité était présent mais insuffisant. En ajoutant des colliers de serrage enserrant les servomoteurs, le gain en rigidité était suffisant pour ne plus constater de jeu.

Cependant, nous n'avons pas trouvé cette solution satisfaisante à long terme. En effet, cela ne permettait pas de démonter la fusée aisément et de la remonter en garantissant un positionnement identique à chaque fois entre les pièces de l'assemblage. Pour pallier ce problème, nous avons fixé une nacelle autour de chaque servomoteur pour garantir la rigidité de l'assemblage. Cette nacelle est présentée figure 5b.

Nous avons décidé de faire ces cadres de renforcement en une pièce séparée du contour de support, comme présentée figure 5, du fait des contraintes de l'impression 3D, que nous avons utilisée pour réaliser ces pièces. En effet, afin d'éviter de devoir recourir à un soutien lors de l'impression qui ajouterait une couche de complexité au processus de fabrication, il est préférable de s'assurer que l'ensemble de la pièce se situe au dessus d'une partie en contact avec le plateau de l'imprimante, donc d'éviter les arches ou les évidements. Dans le cas de la nacelle prévue pour les servomoteurs, la solution que nous avons choisie est donc de réaliser deux pièces encastrables à coller.



(a) Pièce principale du support TVC

(b) Nacelle amovible destinée au maintien des servomoteurs

Figure 5: Design du nouveau support TVC en plusieurs pièces

Nous avons également réimprimé les supports des antennes de l'IMU, comme illustré figure 6. En effet, celles-ci fonctionnent de manière optimale lorsque leur écartement est maximal. Nous avons donc allongé la pièce supportant ces antennes. De plus, les nouvelles antennes que nous avons reçu ont un diamètre inférieur aux précédentes. Cela s'est donc traduit sur la pièce support.

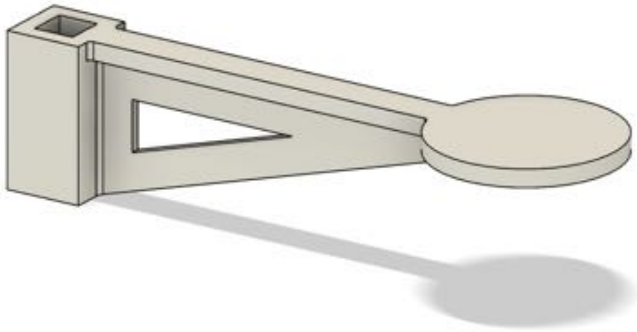


Figure 6: Support des antennes de la centrale inertielle

2 Mesure des performances de la fusée

Afin de faire correctement fonctionner l'algorithme de contrôle, certaines données expérimentales sont indispensables. En particulier, la mesure du couple exercé par la déviation du flux d'air par le TVC en fonction de l'angle des ailettes est nécessaire à l'élaboration d'un modèle utilisable.

Nous avons conçu un banc de test pour faire ces mesures. Cette partie présente la conception de cette expérience, puis les résultats obtenus.

2.1 Travaux préliminaires

2.1.1 Première expérience : vérification de la poussée du moteur

Une des problématiques majeures traitées par le groupe qui nous a précédés a été le fait que la poussée du moteur était trop faible par rapport aux valeurs attendues, à tel point que la fusée ne pouvait pas décoller. Nous avons donc, après avoir hérité du projet, vérifié que cette poussée était suffisante en effectuant un test captif. Nos objectifs étaient les suivants :

- Vérifier que la poussée fournie par le moteur permet bien de faire décoller la fusée ;
- Estimer l'autonomie en vol de la fusée afin de vérifier la pertinence du choix des batteries.

L'expérience a consisté à contraindre les mouvements de la fusée à l'aide de ficelles tendues par les membres du groupe et attachées à la fusée afin de conserver uniquement le degré de liberté de translation verticale. La poussée du moteur, contrôlée en direct à distance par un des membres du groupe, a été progressivement augmentée de manière à garder un comportement stable et éviter des imprécisions dues à des effets dynamiques. La sécurité a été garantie en utilisant des ficelles suffisamment longues, en ne laissant pas la fusée s'élever au dessus des têtes des membres présents, et en faisant usage de casques antibruit.

L'expérience a été concluante. Dès 70 % de puissance du moteur, la poussée compensait le poids de la fusée, ce qui se traduisait par un gain d'altitude de la fusée. C'est donc cette puissance que nous devons considérer pour les estimations d'autonomie en vol. À cette puissance, la batterie 22,2 V met entre 60 et 90 s à s'épuiser.

2.1.2 Premier prototype de banc de test

Ceci fait, nous avons cherché à construire le banc de test afin de recueillir les premiers résultats. Nous avons commencé nos travaux par un prototype préliminaire à la construction d'un banc d'essai plus fiable.

Il a été décidé de le construire avec du bois de récupération et des pièces fournies par le Drahi'X (roulement à bille et tige en métal). Un schéma de ce prototype est présenté figure 7 et des photographies du dispositif sont présentées figure 8. Le dispositif permet de laisser libre la rotation sur un seul axe. Nous mesurons alors le moment sur cet axe en condition

statique. A cet effet, nous avons contacté le TREX de Mécanique pour qu'ils nous fournissent un dynamomètre. Ce capteur à jauge de contrainte est connecté à un ordinateur, il fournit une tension proportionnelle à la force appliquée. Nous avons construit un support en bois pour le placer dans notre dispositif. Afin de déduire la loi reliant la tension fournie par le capteur en fonction de la force appliquée, il est nécessaire de d'abord effectuer une calibration avec un témoin de poids connu. Initialement, nous avons pour cela utilisé un bécher de 100 ml, vide puis rempli d'eau, en considérant un comportement affine pour le capteur. Néanmoins, cette calibration n'a pas été suffisante et a été revue dans le dispositif final.

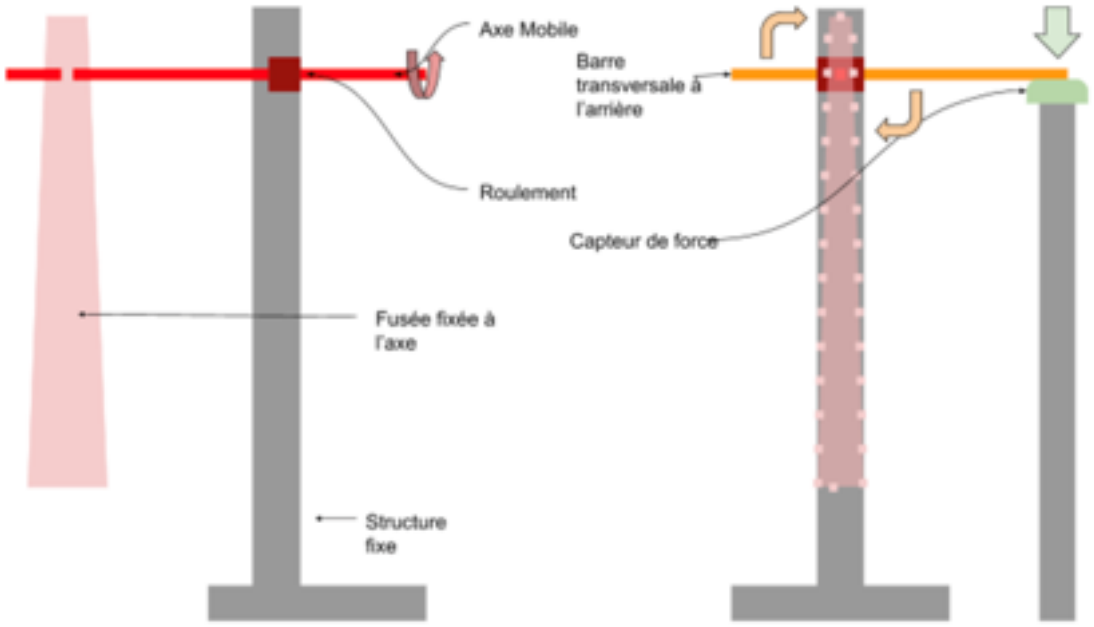


Figure 7: Schéma du dispositif préliminaire

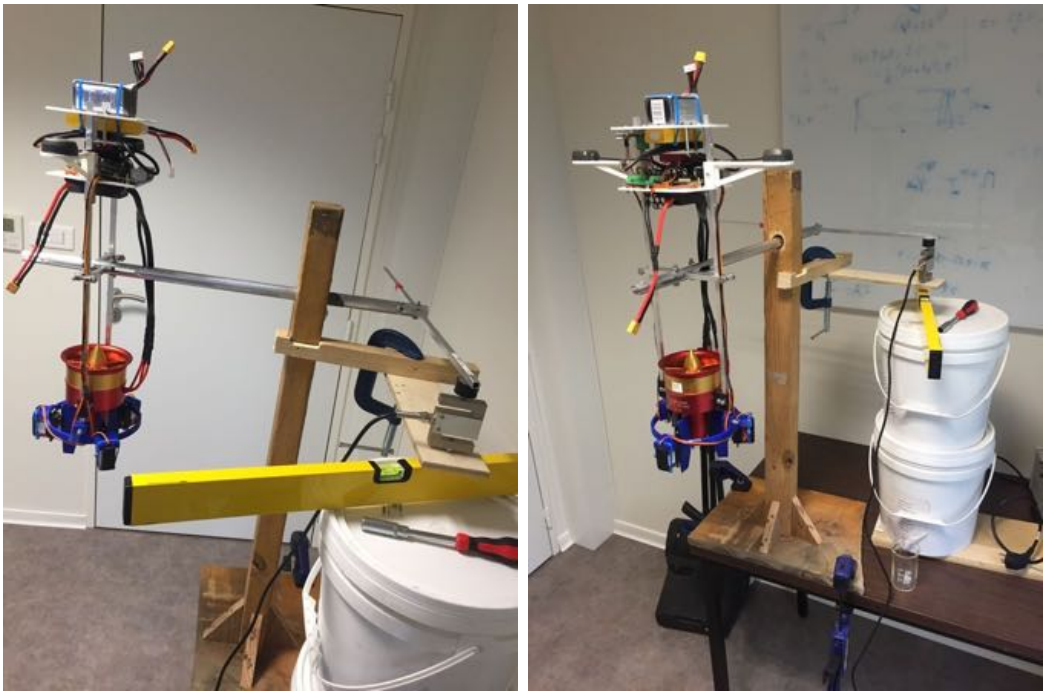


Figure 8: Photographies du dispositif préliminaire

Pour le protocole, les résultats théoriques disponibles en annexe B nous confortent dans l'idée que le comportement des ailettes est linéaire pour de petits angles. Plus précisément, le moment de force appliqué sur la fusée par le TVC dépend linéairement de l'angle des ailettes. Nous choisissons donc de mesurer, pour des angles de 0° à 15° sur une ailette et une puissance moteur de 40 % à 100 % par pas de 10 % le moment exercé par la déviation du flux d'air sur un seul axe de la fusée. Cette mesure permettra ensuite de déduire une approximation du comportement global de la fusée lorsque l'angle des ailettes reste faible.

En dessous de 40 % de la puissance du moteur, la poussée ne compense pas les poids et la fusée retombe violemment. C'est donc un domaine d'usage du moteur non valide dans le cadre d'un vol quasi-statique. De même, au-delà de 15°, l'hypothèse des petits angles est invalide, et le flux d'air devient trop turbulent pour fournir une réponse appropriée au contrôle de la trajectoire. Ce protocole restera donc valable pour l'expérience menée dans le nouveau banc de test.

Les résultats expérimentaux ont été très insatisfaisants avec le prototype de banc de test. Dans un premier temps, nous avons envisagé d'effectuer la mesure durant 10 secondes à une puissance définie, d'attendre au repos, et de recommencer pour une autre puissance. Le passage du moteur de 0 % à 70 % crée un fort effet de choc qui rend les données inexploitables.

Nous avons donc décidé de prendre les mesures en passant par différents paliers sans éteindre le moteur. Le temps passé à puissance constante est arbitraire et contrôlé en direct par les expérimentateurs. De cette façon, nous attendons que le comportement de la fusée soit sta-

bilisé avant de passer à la puissance supérieure. Les résultats sont présentés figure 9.

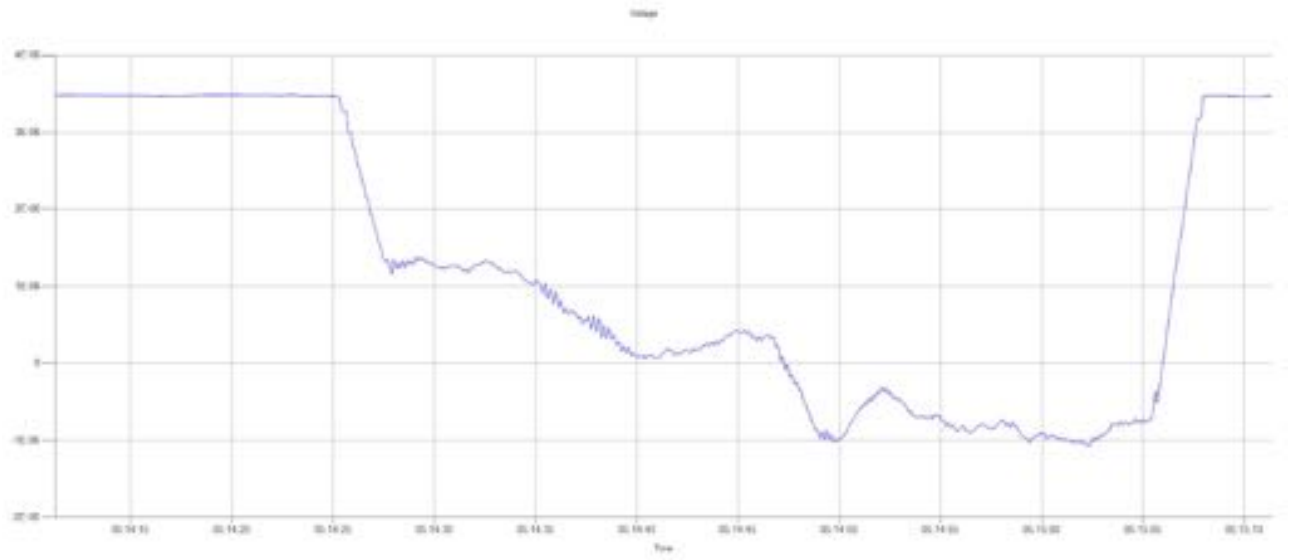


Figure 9: Résultats obtenus - Tension en fonction du temps

On constate une très forte instabilité, quelques phénomènes de résonance, et globalement des résultats loin de nos attentes. Nous attendions en effet quatre paliers bien définis pour les quatre différentes puissance du moteur. Cela n'est absolument pas le cas et aucune loi linéaire ne semble se profiler en fonction de la puissance du moteur, qui en plus de cela n'est pas directement corrélée avec la vitesse du flux d'air.

La solution que nous avons choisie pour pallier ce problème de données expérimentales inutilisables est la conception d'un nouveau banc de test en profilés aluminium, lesquels sont beaucoup plus adaptés que le bois pour un montage précis et une réduction maximale du bruit de mesure engendré par les vibrations du banc. Nos premières expérimentations ont en effet montré que les effets de résonance qui polluaient nos résultats provenaient d'une trop grande souplesse du premier banc construit en bois.

2.2 Cahier des charges

L'expérience cherche à mesurer les contraintes exercées sur la fusée par le TVC. Une fois celle-ci placée sur un axe fixe, nous mesurons le moment appliqué lorsqu'une ailette du TVC est écartée de sa position neutre. Nous récoltons ces données en fonction de l'angle de l'ailette et de la puissance du moteur. Nous avons fait le choix de réduire le problème sur un seul axe de tangage, puis nous utiliserons les symétries du dispositif pour en déduire une loi sur l'ensemble des axes de tangage et d'inclinaison.

Néanmoins, nous avons eu l'ambition de construire un banc de test plus complexe qui pourrait être utilisé pour bien d'autres expériences que l'unique mesure du moment d'action

d'une ailette sur un axe de rotation. Nous avons souhaité un banc permettant des conditions d'expérience s'approchant au maximum des conditions réelles de vol, c'est-à-dire avec le plus grand nombre possible de degrés de liberté.

En particulier, il doit permettre 3 degrés de liberté en rotation, afin de pouvoir effectuer des tests dans les conditions les plus proches possibles du vol libre. Cela permet également de tester tous les degrés de liberté en rotation individuellement en bloquant les rotations non désirées, sans devoir reconstruire un banc d'essai à chaque fois.

Une étude théorique a été menée en amont de l'expérience. Elle a permis de valider la cohérence du projet, et d'avoir une attente vis-à-vis des résultats. C'est donc une loi linéaire en l'angle de l'ailette et en la vitesse du flux d'air que nous attendons.

2.3 Conception du banc de test

Afin d'obtenir la meilleure précision possible lors des tests effectués avec le banc d'essai, nous avons opté pour une structure formée de profilés extrudés, qui présentent des avantages économiques et pratiques puisqu'ils sont aisés à assembler. Les pièces de cette structure étant standards, exception faite aux quelques pièces que nous avons conçues et fabriquées par impression 3D, les sources d'imprécision sont grandement réduites par rapport au premier prototype en bois.

Le degré de liberté en rotation dans l'axe de la fusée (axe x vertical) est obtenu en positionnant celle-ci au centre d'un roulement à billes présentant un évidage assez large pour accepter la fusée.

Ce roulement est lui-même fixé à un cadre orientable grâce à deux jeux de roulements d'axes perpendiculaires. Cet assemblage permet d'obtenir, entre la fusée et la structure du banc d'essai, une liaison sphérique. Cette structure est mise en évidence figure 10.

Une telle architecture présente cependant un inconvénient principal. En effet, le moment d'inertie de l'ensemble composé de la fusée et de son support n'est pas équivalent au moment d'inertie de la fusée seule. Toutes les parties mobiles du banc de test interviennent dans cette modification du moment d'inertie, et comme ces parties ont des masses non négligeables devant la masse de la fusée, l'influence sur le moment d'inertie est forte. Par conséquent, on peut s'attendre à ce que la fusée se comporte différemment lors d'un essai sur le banc de test que lors d'un vol libre. En effet, le moment d'inertie intervient dans les équations décrivant le comportement dynamique de la fusée, comme expliqué en annexe B.

Cependant, cet inconvénient n'est pas contraignant pour les essais que nous envisageons. En effet, les expériences les plus importantes et nécessaires au bon paramétrage des algorithmes de contrôle, en particulier l'expérience décrite en section 2.1.2, sont des expériences statiques, au cours desquelles la fusée reste immobile. Les degrés de liberté permettent alors non pas à la

fusée de s'orienter, mais de pouvoir mesurer le couple exercé par le TVC sur la fusée en fixant un dispositif de mesure. Ce ne sera alors pas la structure qui bloquera la rotation de la fusée mais le dispositif de mesure. Ces expériences étant réalisées en régime statique, le moment d'inertie n'intervient pas, donc l'inconvénient précité n'est pas rédhibitoire.

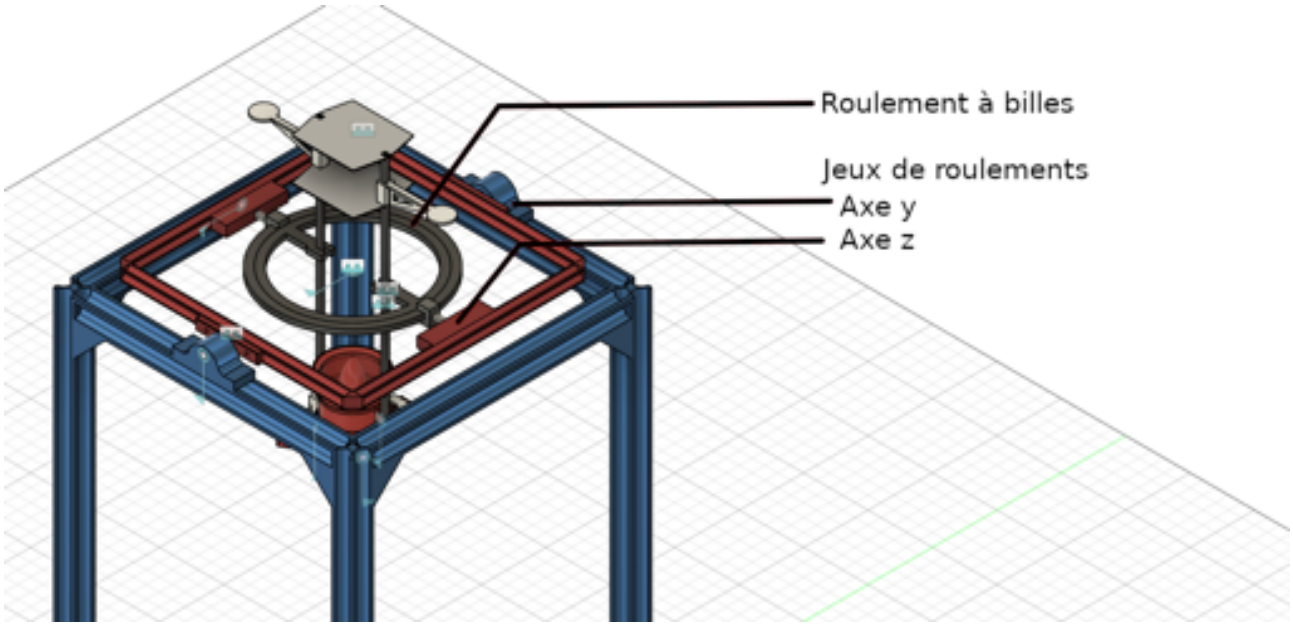


Figure 10: Nouveau banc d'essai

Une des problématiques rencontrées lors de la conception de ce banc de test a été la fixation de la fusée au roulement à billes. Celui-ci consiste en deux couronnes de 12 mm de largeur et 8 mm d'épaisseur, présentant chacune 6 trous de 5 mm destinés à faire passer des vis. Nous avons donc dû concevoir une pièce répondant à ces exigences, présentée figure 11. Afin de conserver une rigidité maximale et de garantir la solidarité de l'ensemble, la pièce n'a pas été réalisée en deux morceaux mais les deux côtés sont joints en leur centre. Au vu de la structure de la fusée, une fixation au niveau des barres verticales nous a paru être la meilleure option. Afin de ne pas devoir démonter la fusée à chaque montage sur le banc de test, nous avons décidé de tenir la fusée par pincement, le serrage étant effectué avec un boulon pour chaque barre verticale. Le pincement permet également de pouvoir repositionner la pièce sur toute la longueur des barres verticales, et donc de la placer au niveau du centre de masse de la fusée.

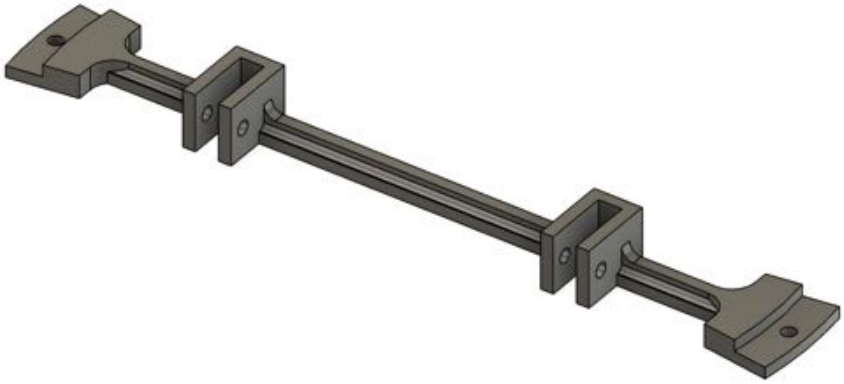


Figure 11: Support de la fusée sur le banc de test

La figure 12 expose le banc une fois monté et le positionnement de la fusée dans celui-ci. Les paliers à roulement sur la structure fixe, au bas de l'image, permettent au cadre mobile plus petit de pouvoir s'orienter selon l'axe y. Les deux roulements placés sur le cadre mobile permettent la rotation de la fusée selon l'axe z perpendiculaire au premier. Enfin, le plateau à roulement dans lequel est fixée la fusée au moyen de la pièce présentée figure 11 permet sa liberté selon l'axe x vertical.

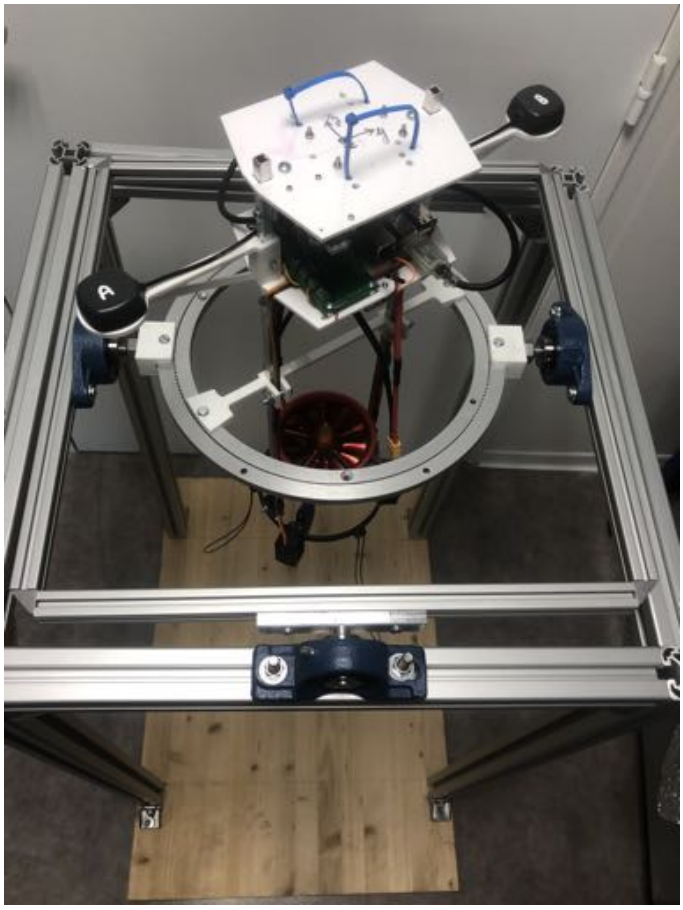


Figure 12: Assemblage final du banc et fixation de la fusée au niveau de son centre de masse

2.4 Mesures et résultats

Il nous reste à effectuer les essais permettant d'acquérir la loi de comportement liant la consigne de position angulaire transmise aux servomoteurs et le couple exercé par le TVC sur la fusée. L'objet de cette partie est de décrire le protocole de ces essais.

2.4.1 Mesure statique de l'action des ailettes

Le nouveau banc d'essai, respectant nos exigences exposées en section 2.2, devrait nous permettre de recommencer, avec, nous l'espérons, une plus grande précision qu'avec le précédent banc de test, les mesures du moment exercé par une ailette sur un seul axe de rotation libre. Ceci nous permettra d'évaluer l'action des ailettes pour compléter l'algorithme de contrôle en inclinaison présenté en section 3.1.2.

Cette expérimentation sera statique puisque le capteur de force bloquera l'axe de rotation étudié pour pouvoir évaluer les contraintes qui lui sont appliquées. Nous ne mesurerons l'action que d'une seule ailette et sur un seul axe de rotation libre, ce qui nous permettra ensuite, grâce aux calculs présentés en annexe B, de reporter l'action de chaque ailette sur chaque axe de

rotation.

Nous reprendrons *a priori* le protocole utilisé lors de la précédente expérience décrite en section 2.1.2 : nous effectuerons la mesure pour des angles d'ailettes variant de 0° à 15° par pas de 1°, et pour une puissance du moteur variant de 40 % à 100 % par pas de 10 %, sur des durées assez longues pour observer un comportement stabilisé.

2.4.2 Validation du contrôle de la fusée captive

Lors de cette seconde phase d'expérimentation, nous utiliserons les trois degrés de liberté permis par notre nouveau banc de test. À ce stade nous souhaiterons observer le comportement de la fusée dans une situation de liberté s'approchant de la réalité afin de valider la première version d'algorithme de contrôle déjà écrite. Ces tests seront dynamiques et l'inertie due au banc de test ajoutée au mouvement de la fusée sera à prendre en compte dans nos observations et conclusions quant au comportement de la fusée. Si ces tests ne correspondront pas tout à fait au vol libre, ils permettront toutefois une première phase de validation, sous contrôle, des algorithmes de la fusée. Nous pourrons, à l'issue de ces essais préliminaires, envisager le vol libre.

3 Contrôle

3.1 Structure de l'algorithme

L'algorithme de contrôle de la fusée prend en entrée les données actuelles de position, vitesse et accélération de la fusée. Il les compare à l'objectif souhaité et fournit en conséquence les consignes aux actionneurs. Le système de contrôle comprend pour cela un régulateur PID (Proportionnel Intégral Dérivé) chargé de réguler la poussée moteur, et un système LQR (*Linear Quadratic Regulator*) chargé de contrôler les ailettes du TVC. L'ensemble du code Python est disponible en annexe C.

3.1.1 Contrôle de la poussée moteur

Nous avons paramétré un régulateur PID, qui contrôle la poussée moteur afin de respecter une consigne d'altitude (coordonnée x selon les conventions de l'aérospatiale). En effet, dans l'approximation des petits angles, on néglige l'influence des ailettes sur la poussée verticale s'appliquant à la fusée. Ainsi, l'accélération verticale (et donc l'altitude) est directement contrôlée par la poussée du moteur. Un réglage minutieux des différents paramètres proportionnel, intégral et dérivé du correcteur PID reste cependant nécessaire comme nous le verrons dans les résultats de simulation en partie 3.2.

Par ailleurs, le correcteur PID ne prend pas en compte les données concernant la dynamique du système, au contraire d'un régulateur de type LQR par exemple. C'est le régulateur que nous avons utilisé pour contrôler le TVC ; nous en détaillerons le fonctionnement dans la partie 3.1.2. Cela le rend plus robuste aux défauts de modélisation mais également moins spécifique au système contrôlé. Par exemple, le poids, qui n'est pas visible à l'instant $t = 0$ car compensé par la réaction du sol, apparaît comme une perturbation constante par la suite, ce qui peut perturber le correcteur PID au moment du décollage. Pour éviter cet effet, que nous avons détecté dans les essais de l'algorithme, nous avons fait une simple modification : au lieu de récupérer directement la poussée moteur F_e en sortie du PID, on considère que c'est la poussée résultante $F_e - Mg$ (poussée moins le poids) qui est fournie en sortie du régulateur. C'est à dire que la consigne de poussée envoyée au moteur est égale au poids plus la valeur en sortie du PID. Ainsi, on augmente graduellement la poussée moteur jusqu'à $F_e = Mg$ au décollage puis le PID n'a plus qu'à faire varier légèrement la poussée résultante pour augmenter ou diminuer l'altitude.

Nous avons d'abord fait une modélisation sur Simulink de ce système-là, qui demeure très classique. Nous avons déterminé les valeurs proportionnelle, intégrale et dérivée du PID à l'aide de l'application *PID Tuner* de Matlab. Le schéma-bloc du modèle est à retrouver dans la figure 13.

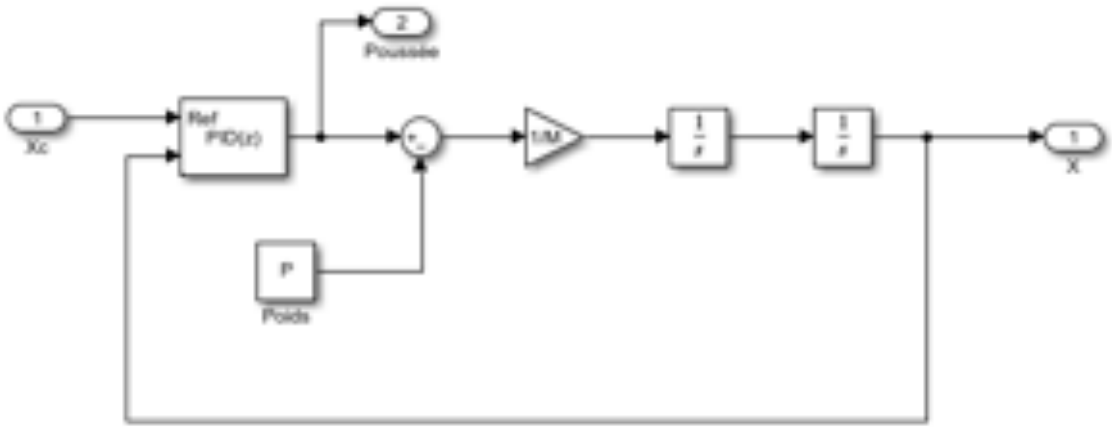


Figure 13: Modèle de l'asservissement sur l'axe x

A partir de ce modèle, nous avons ensuite travaillé sur l'implémentation Python de ce régulateur afin de l'intégrer à la fusée par la suite. Elle est désormais fonctionnelle. Nous avons pour cela utilisé le package "simple_pid" implémentée par Martin Lundberg [5]. Le code python est disponible en annexe C.

3.1.2 Contrôle du TVC : inclinaison des ailettes

Une fois l'altitude contrôlée de son côté, il reste encore à contrôler les deux axes du plan horizontal (y et z selon les conventions du secteur spatial) ainsi que les 3 angles (tangage θ, lacet ψ et roulis ϕ). Ce contrôle est effectué via le TVC de la fusée, c'est-à-dire 3 ailettes actionnées par des servo-moteurs qui orientent le flux du moteur.

Initialement, le groupe de la promotion X2018 avait prévu de contrôler ces ailettes via un régulateur PID complémentaire à celui qui contrôle la poussée moteur. Ils se basaient en effet sur deux moteurs qui, étant contra-rotatifs, ne produisaient aucun couple autour de l'axe vertical de la fusée. Cependant, l'architecture de la fusée a dû être revue en abandonnant le principe des deux moteurs contra-rotatifs. Ce changement rend obligatoire la capacité à contrôler l'angle de roulis (selon l'axe vertical) car désormais le moteur (unique) applique un couple non nul à la fusée. Ce nouveau paramètre, ajouté à des erreurs de réglage du PID des X2018, a poussé le groupe X2019 à se tourner vers un régulateur linéaire quadratique, désigné LQR dans la suite (Linear Quadratic Regulator). Ce dernier est plus adapté au contrôle d'un grand nombre de variables (ici la position selon deux axes, trois angles, ainsi que les vitesses correspondantes). Il prend de plus en compte la dynamique du système, permettant un contrôle plus fin [6].

Les contrôleurs LQR sont utilisés dans le cas de systèmes dynamiques décrits par un ensemble d'équations différentielles linéaires et dont le coût est décrit par une fonction quadratique. La fonction de coût est souvent définie comme la somme des écarts des paramètres clés, comme

la position ou l'angle de la fusée et des ailettes par rapport à leurs valeurs souhaitées.

Cet algorithme LQR permet de calculer la matrice de gains K à insérer dans la boucle de retour pour obtenir un contrôleur par retour d'état. Pour l'obtenir, il utilise des facteurs de pondérations que nous devons lui fournir dans la matrice Q et le vecteur R. La loi de retour d'état $U = -KW$ minimise la fonction de coût quadratique $J(U) = \int_0^\infty (W^T Q W + U^T R U)\,\mathrm{d}t$ tel que la dynamique du système est décrite par $\dot{W} = AW + BU$ avec W vecteur d'état de la fusée, U vecteur des paramètres de contrôles (ici les 3 angles des ailettes) et A et B deux matrices [7]. Pour choisir Q et R il faut déterminer ce qui est prioritaire dans le contrôle de notre système. Q attribue un coût à l'erreur sur chaque variable d'état. Plus ce coût est élevé, plus le LQR va chercher à diminuer l'erreur correspondante. De même, R attribue un coût à l'utilisation de chaque variable de contrôle. Plus le coût est élevé, moins le LQR va solliciter ce contrôle. Le réglage de ces variables de coût est essentiel au bon fonctionnement du LQR, comme nous le verrons dans la partie 3.2.2.

Afin d'implémenter ce régulateur dans notre fusée, nous avons, comme avec le contrôle de la poussée moteur, d'abord modélisé le système sur Simulink. Un schéma général du contrôle selon les axes y et z et le concept d'élaboration de la consigne à envoyer aux ailettes du TVC est détaillé en figure 14.

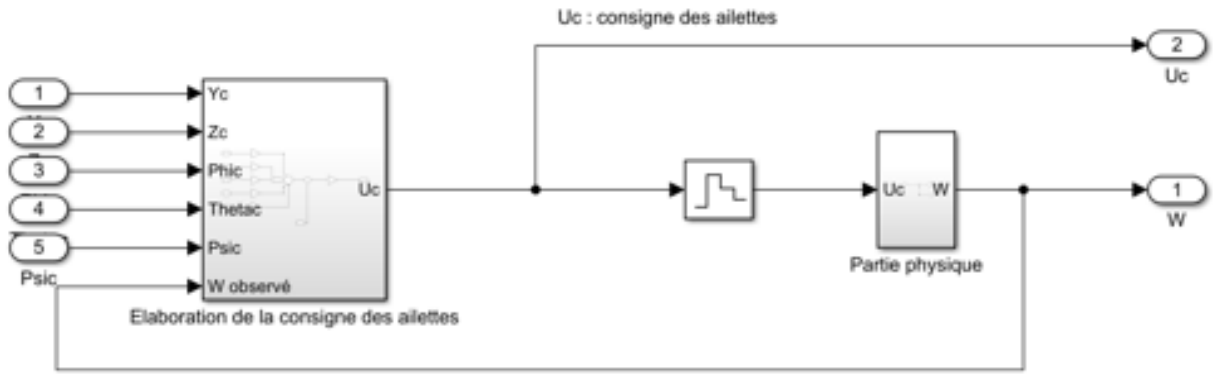


Figure 14: Modèle de l'asservissement sur les axes y et z

Ce schéma-bloc est divisé en plusieurs parties. D'abord, la partie physique est présentée en figure 15. Il s'agit de la traduction en schéma-blocs de l'application du principe fondamental de la dynamique et du théorème fondamental de la dynamique détaillé en annexe B. $U_c = \begin{bmatrix}\alpha_1^c & \alpha_2^c & \alpha_3^c\end{bmatrix}^T$ est le vecteur des consignes d'orientation des ailettes, qui engendre alors un vecteur d'état de la fusée $W = \begin{bmatrix}y & \dot{y} & z & \dot{z} & \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi}\end{bmatrix}^T$.

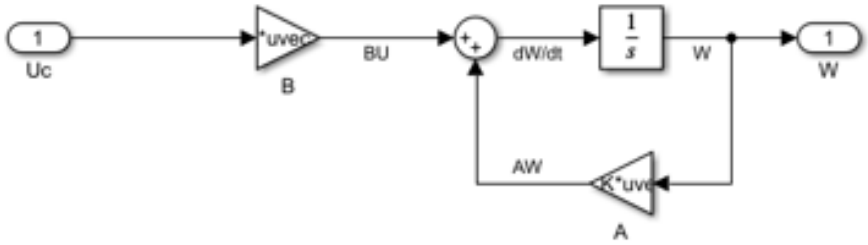


Figure 15: Modélisation de la physique du système

À partir de l'état de la fusée W et des consignes en position y_c et z_c et en angles ϕ_c, θ_c et ψ_c, le bloc "Élaboration de la consigne des ailettes" permet de déterminer U_c. C'est ici que le régulateur LQR entre en jeu, afin de corriger l'erreur entre la consigne et les données. Le schéma-bloc de cette partie est donné en figure 16.

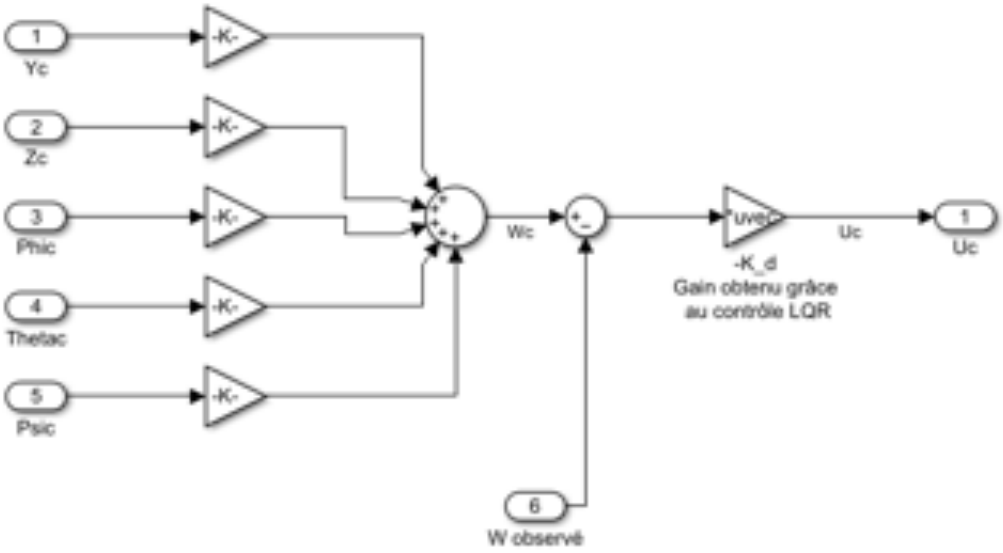


Figure 16: Modélisation de l'élaboration de la consigne aux ailettes

La partie de gauche est simplement la construction d'un vecteur à partir des consignes. L'élaboration de la consigne consiste, comme expliqué plus tôt, en l'application du LQR avec $U = -K_d W$. Ce gain est calculé grâce à un module Matlab, et accessible grâce au code donné en figure 17.

```
[K_d,S_d,e_d] = dlqr(Ad,Bd,Q,R);
```

Figure 17: Code nécessaire à la mise en place d'un contrôle LQR

Le groupe précédent n'avait eu le temps que d'essayer ce module. Nous avons repris la théorie de ce type de contrôle pour l'implémenter sur Python, voir annexe C. Par ailleurs, il est important de faire la jonction entre le contrôle des ailettes et le contrôle de la poussée moteur. Si dans l'approximation des petits angles, les ailettes n'influent pas sur le comportant du PID, l'inverse n'est pas vrai : la poussée moteur influe directement sur le comportement du TVC et donc du LQR. C'est une problématique qui avait été éludée l'année dernière, les simulations du LQR avaient alors été réalisées en considérant la poussée moteur constante égale au poids de la fusée. Pour améliorer le comportement global du système de contrôle de la fusée, nous avons donc pris en compte cette problématique dans notre implémentation. Les deux algorithmes (PID et LQR) ont ainsi été réglés conjointement. Le facteur le plus important a été de veiller à ce que la consigne de poussée moteur fournie par le PID reste proche du poids en ne variant pas de façon trop abrupte, pour laisser le temps au LQR de s'adapter au fur et à mesure. Les résultats de ces paramétrages sont donnés en 3.2.2.

3.1.3 Filtre de Kalman

Les données qui sont utilisées par les algorithmes de contrôles sont celles fournies par l'IMU. Malgré sa précision, il existe toujours des incertitudes de mesure. Nous en avons notamment discuté avec Antoine Tavant, directeur du Centre Spatial de l'École polytechnique, qui nous a recommandé l'utilisation d'un filtre de Kalman [8]. Ce dernier utilise une connaissance partielle de la dynamique du système et des incertitudes de mesures pour fournir une estimation optimale de la donnée mesurée. Supposons qu'on connaît l'état de la fusée à l'instant t (position, vitesse et accélération). On peut alors effectuer une prédiction de l'état à l'instant $t+1$ grâce au modèle dynamique de la fusée. Cependant cette prédiction est approximative car le modèle ne peut pas être parfait par nature, sans compter les éventuelles perturbations extérieures. Une autre possibilité est de directement mesurer grâce à l'IMU l'état de la fusée à l'instant $t+1$, mais on a alors des incertitudes de mesure.

En combinant ces deux sources d'information, on peut obtenir une meilleure estimation du véritable état à l'instant $t+1$. Il faut attribuer un poids à chacune des sources en fonction de la crédibilité qu'on lui apporte. Si le modèle est très approximatif et le capteur très précis, on accorde plus de poids aux données du capteur. À l'inverse, si le modèle est très complet et fiable mais que le capteur est peu précis, on accorde plus de poids aux prévisions des calculs de dynamique. C'est un raisonnement qu'il faut bien sûr appliquer séparément à chaque donnée mesurée $(x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi})$ car les incertitudes des capteurs de positions, angles ou vitesses ne sont pas les mêmes. La première méthode de filtre envisageable est d'attribuer un poids constant à chacune des sources (modèle dynamique ou capteur). Ce type de filtre est appelé alpha-beta [9].

Le filtre de Kalman, décrit en 1960 par R.E. Kalman [8], permet d'attribuer des poids optimaux calculés à chaque étape pour minimiser la variance de l'estimation. Il s'agit d'un des filtres les plus utilisés dans le domaine du contrôle, y compris dans le secteur spatial où il a notamment servi dans le programme Apollo. De nombreux travaux ont été menés à son sujet,

ce qui nous a facilité l'assimilation de la théorie associée [10]. Un projet collaboratif ayant pour objectif la vulgarisation des travaux de Kalman nous a notamment permis d'implémenter notre propre version du filtre de Kalman sans reposer sur des librairies externes [11]. Le code est disponible en annexe C. Les opérations que mènent le filtre de Kalman sur les données sont résumées en figure 18.

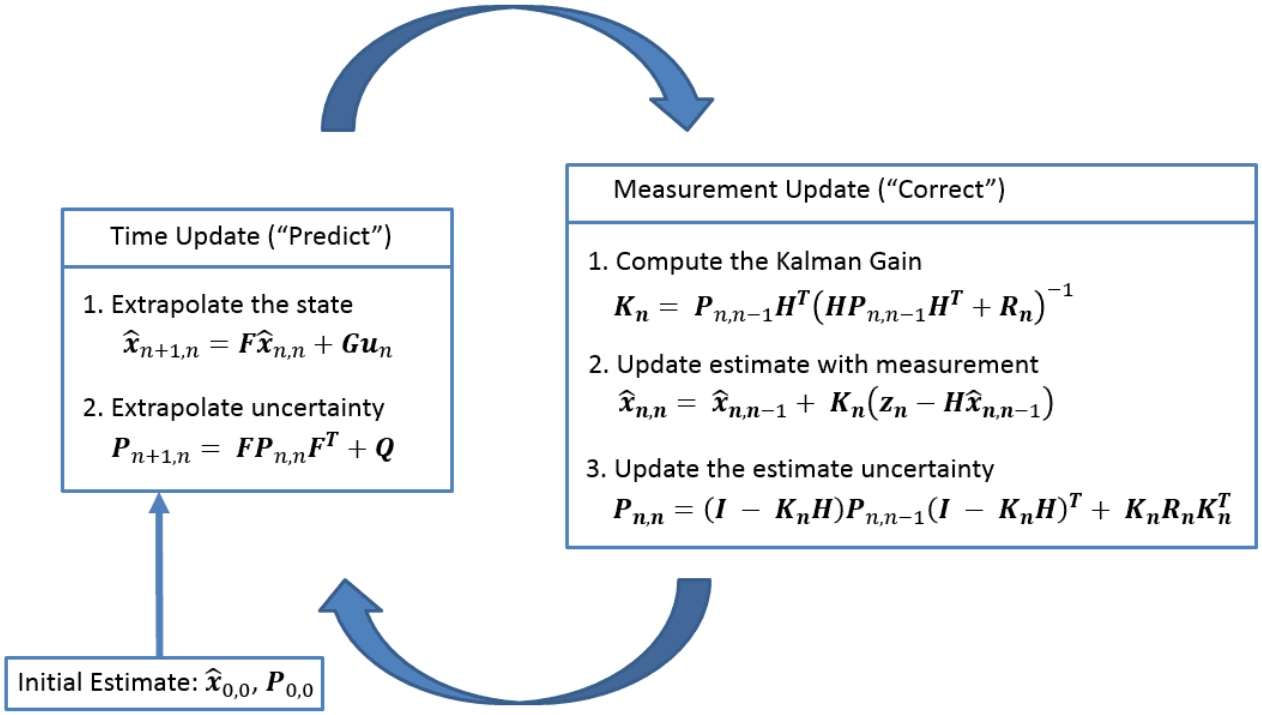


Figure 18: Diagramme du filtre de Kalman

Terme	Nom
x	Vecteur d'état
u	Vecteur de contrôle
F	Matrice de transition d'état (matrice A du LQR)
G	Matrice de contrôle (matrice B du LQR)
P	Matrice de covariance de l'erreur
R	Matrice de covariance du bruit de mesure
z	Vecteur de mesure
H	Matrice d'observation
Q	Bruit de processus

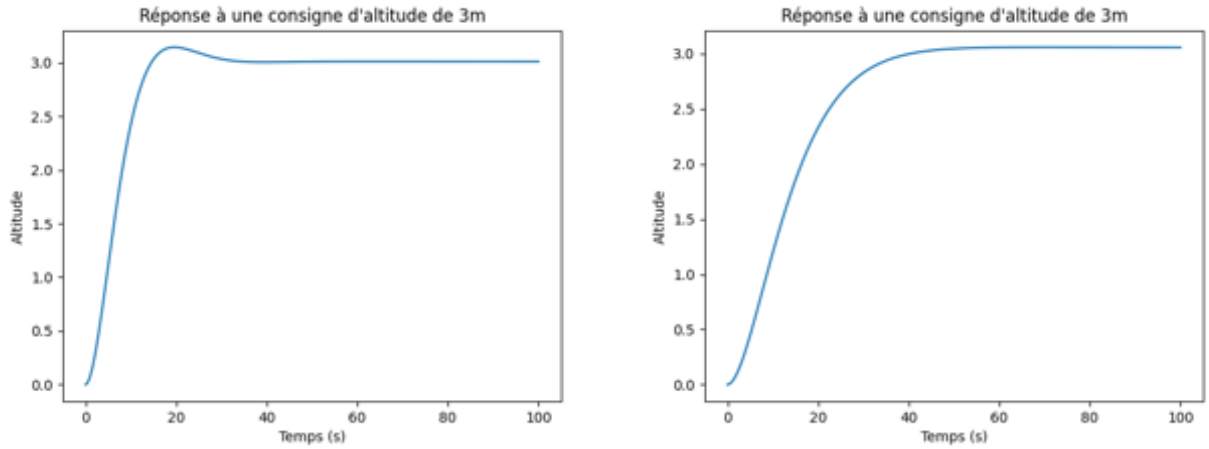
Les termes x, u, F et G ont déjà été définis lors de l'implémentation du LQR. Les matrices P, R et Q seront déterminées en partie 3.2.3. Attention à ne pas confondre les matrices Q et R du filtre de Kalman avec celles du LQR. Le vecteur z est celui des mesures, directement renvoyé par l'IMU. La matrice d'observation H est telle que $z = Hx$ or dans notre cas l'IMU nous permet de mesurer l'ensemble de nos variables d'état, ce qui donne simplement $H = I_{10}$.

3.2 Simulations et réglages

Les algorithmes de contrôle ainsi que le filtre de Kalman reposent sur certains paramètres qu'il est nécessaire de déterminer de façon empirique pour de meilleurs résultats. Pour cela, après avoir effectué l'implémentation en Python, nous avons fait passer une batterie de tests et de simulations à nos algorithmes pour déterminer le meilleur paramétrage pour obtenir des réponses robustes, suffisamment rapides mais qui ne sollicitent pas trop les actionneurs.

3.2.1 Essais du régulateur PID

Le réglage du régulateur PID a nécessité un paramétrage très précis de ses composantes pour obtenir de bonnes performances. Nous avons employé différentes méthodes classiques de réglage de régulateur PID [12] [13]. L'utilisation de Python nous a permis d'effectuer des milliers d'essais différents en faisant varier nos paramètres proportionnel, intégral et dérivé sur différentes trajectoires. Notre objectif était d'obtenir une réponse qui soit suffisamment lisse, peu oscillante, sollicitant peu le moteur tout en permettant un temps de réponse raisonnable de quelques dizaines de secondes pour une consigne de 3 mètres d'altitude. Les différentes méthodes employées nous ont permis d'obtenir deux types de réglages satisfaisants et robustes, qui sont présentés en figure 19.



(a) Réponse rapide

(b) Réponse sans dépassement

Figure 19: Réponse pour deux types de réglages du régulateur PID

Pour décrire les réglages nous noterons P la correction proportionnelle, I la correction intégrale et D la correction dérivée. Le réglage correspondant à la figure 19a est $P = 7 \times 10^{-2}, I = 10 \times 10^{-4}, D = 1.0$. Celui correspondant à la figure 19b est $P = 2 \times 10^{-1}, I = 10 \times 10^{-4}, D = 1.25$. Les deux disposent d'un important correcteur dérivé et d'un correcteur intégral très faible. Pour comprendre ces réglages, il faut revenir sur le principe d'un correcteur PID. Dans le contrôle proportionnel, le facteur de correction est déterminé par l'ampleur de la différence entre le point de consigne et la valeur mesurée. Le problème avec cette approche,

c'est qu'à mesure que la différence se rapproche de zéro, la correction se rapproche aussi de zéro. Par conséquent, l'erreur n'atteint jamais zéro. Cela peut par exemple être le cas pour un système de chauffage.

La fonction intégrale résout ce problème en tenant compte de la valeur cumulative de l'erreur. Plus la différence entre le point de consigne et la valeur réelle persiste, plus l'ampleur du facteur de correction calculé est grande. Cependant, un retard de réponse à la correction conduit à un dépassement et parfois à une oscillation par rapport au point de consigne. C'est précisément le cas de notre système. Une fusée en vol réagit fortement aux sollicitations et, du fait de son inertie, est peu sujette à la problématique d'une correction tendant vers 0 à mesure qu'on se rapproche de la valeur consigne. Au contraire, une correction intégrale trop importante pourrait entraîner des oscillations, ce que l'on souhaite éviter.

La fonction dérivée à l'effet inverse : elle examine le taux de changement en cours et modifie progressivement le facteur de correction pour en atténuer l'effet au fur et à mesure que la valeur réelle se rapproche de la valeur de consigne. C'est pourquoi sa valeur est assez importante dans notre cas : il faut diminuer la poussée moteur avant d'arriver à la valeur consigne pour éviter un dépassement trop important dû à l'inertie.

Le premier réglage envisagé en figure 19a a l'avantage d'être assez rapide, avec un temps de réponse d'environ 35s. Il présente un léger dépassement mais pas d'oscillations. Si le dépassement n'est pas problématique lors de la phase d'ascension, ce n'est bien sûr pas le cas lors de l'atterrissage de la fusée. Nous avons donc travaillé à établir un nouveau réglage sans dépassement, en augmentant le correcteur dérivé. Celui-ci présente un temps de réponse un peu plus long, d'environ 45s, mais cela reste raisonnable au vu de l'autonomie de la fusée. Son approche en "douceur" de la valeur consigne permettra notamment d'assurer un atterrissage sans encombre à la fusée. Nous avons également mesuré la variation maximale de poussée moteur par rapport au poids, fournie par le régulateur PID : 0.6N pour la version avec dépassement, 0.2N pour la version sans dépassement. Le poids de la fusée étant d'environ 40N, on a donc bien une faible variation de la poussée moteur autour du poids de la fusée. C'était l'un de nos objectifs, pour assurer une bonne coordination avec le LQR et le TVC.

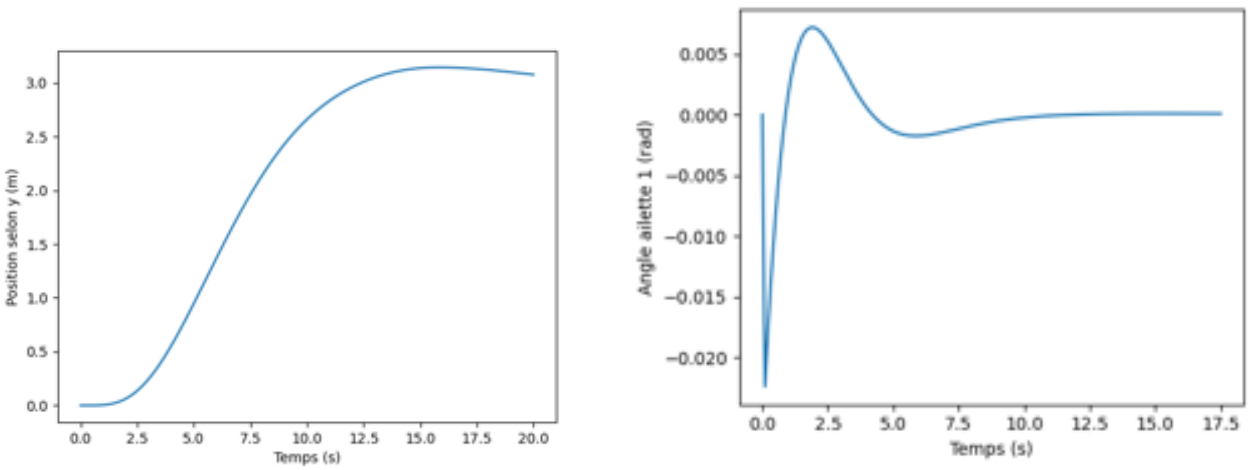
3.2.2 Essais du LQR

Une fois le régulateur PID testé et réglé, nous avons pu nous attaquer au réglage du LQR, chargé du contrôle des ailettes du TVC et donc des variables $y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi$ et $\dot{\psi}$. Déterminer la matrice de coût Q et le vecteur de coût R sont en effet déterminant sur le comportement de l'algorithme de contrôle LQR. Q est une matrice de taille 10×10 qui détermine le coût de l'écart à la consigne pour chaque variable. Autrement dit, Q détermine les variables pour lesquelles se rapprocher de la consigne au plus vite est prioritaire. Les valeurs hors diagonales sont celles d'un éventuel coût croisé des variables. On pourrait par exemple accorder peu d'importance à l'écart sur y ou z séparément mais vouloir éviter un écart simultané sur y et z. C'est une situation très particulière et *a priori* pas notre cas donc nous prendrons

Q diagonale.

On peut alors distinguer deux types de variables : les variables de position et les variables angulaires. On prendra de ce fait une valeur $p \in \mathbb{R}_+$ pour les 4 premiers éléments diagonaux et $a \in \mathbb{R}_+$ pour les 6 suivants. R quant à lui est un vecteur de taille 3 qui détermine le coût de sollicitation des actionneurs, ici les 3 ailettes. Étant donné que l'utilisation d'une ailette particulière n'est pas plus coûteuse qu'une autre, il est clair que R est de la forme $R = r \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ avec $r \in \mathbb{R}_+$.

Notre modèle dynamique de la fusée est basé sur l'hypothèse des petits angles : on suppose que l'on va maintenir la fusée aussi verticale que possible tout au long de son déplacement. Il faut donc $a >> p$. Il faut de plus donner un coût important à l'utilisation des ailettes pour éviter que les angles envoyés par le LQR sortent de cette approximation des petits angles. En effet, supposons que l'on choisisse r petit devant p, le LQR va prioriser le respect de la consigne en sollicitant fortement les actionneurs. Dans notre cas, cela voudrait dire demander de grands angles aux ailettes pour un déplacement latéral rapide. C'est ce qu'on le souhaite éviter car cela briserait l'équilibre de la fusée. Il faut donc $r >> p$. Une fois ces relations établies, la valeur exacte peut se déterminer via les essais et simulations que nous avons menés. En prenant $a = r = 10 \times 10^3$ et $p = 10 \times 10^{-1}$ nous obtenons le résultat de la figure 20a pour la position selon l'axe y en fonction du temps.



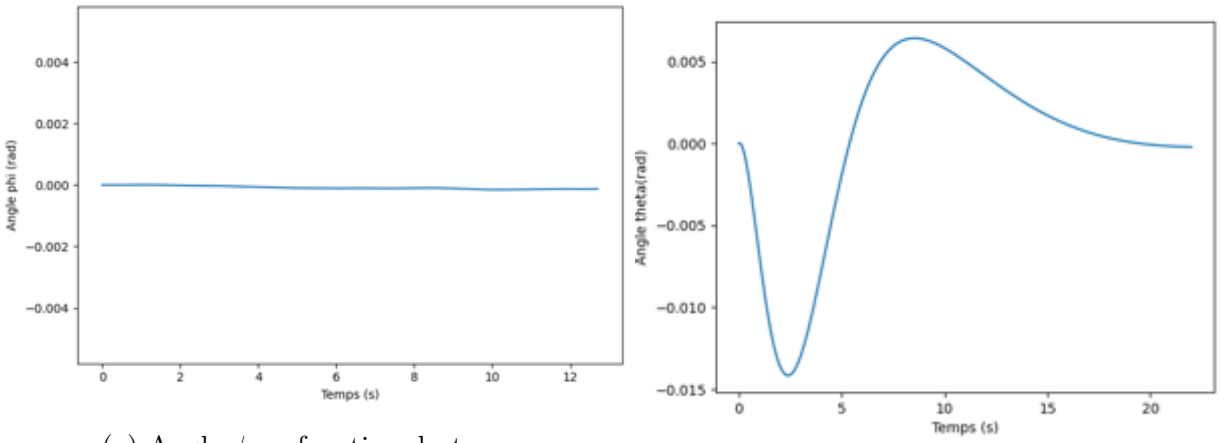
(a) Position selon l'axe y en fonction du temps

(b) Angle de l'ailette 1 en fonction du temps

Figure 20: Position selon y et sollicitation correspondante des ailettes

On observe un temps de réponse d'une vingtaine de secondes sans oscillations ce qui correspond à nos objectifs. L'évolution de la coordonnée z est très similaire. De plus lors de ce déplacement l'angle des ailettes, visible en figure 20b reste inférieur à 0.025 rad c'est à dire 1,5° ce qui est dans le domaine de validité de l'approximation des petits angles. La sollicitation

des ailettes 2 et 3 présente une allure très similaire. On peut également tracer la valeur de deux angles : ϕ (roulis) qui doit rester quasiment nul et θ (tangage) qui doit varier légèrement pour permettre le déplacement latéral mais rester très petit devant 1 si notre paramétrage est correct. Le résultat est visible en figure 21.



(a) Angle ϕ en fonction du temps

(b) Angle θ en fonction du temps

Figure 21: Réponse angulaire de la fusée

On observe effectivement que l'angle de roulis reste très proche de 0 tandis que l'angle de tangage varie autour de 0 pour permettre le déplacement latéral. La valeur absolue de ce dernier ne dépasse pas 0,015 radians soit 0,86 degrés, on reste donc bien dans le domaine de validité de l'approximation des petits angles. Les résultats sont similaires pour l'angle ψ (lacet).

3.2.3 Essais du filtre de Kalman

Une fois les algorithmes de contrôle en place, il s'agit de paramétrer le filtre de Kalman. La matrice de de covariance de l'erreur P est recalculée à chaque itération et ne nécessite que d'être initialisée. À l'instant $t = 0$ la fusée est à l'origine du repère, par convention, donc l'erreur est rigoureusement nulle. On prendra cependant $P = 10 \times 10^{-4} \times I_{10}$ car le processus d'itération nécessite une matrice de départ non nulle. La matrice de covariance du bruit de mesure est constante. Comme chaque variable est mesurée indépendamment par l'IMU on peut la prendre diagonale. Chaque élément diagonal est alors la variance de la mesure si on considère que le bruit est gaussien. Cette valeur peut être directement mesurée en observant l'amplitude des oscillations d'une mesure alors que la fusée est statique.

Il ne reste ainsi plus qu'à déterminer la matrice du bruit de processus (process noise), ce qui est plus difficile que pour le bruit de mesure. Le "process noise" est en effet une façon de quantifier les incertitudes du modèle dynamique (qui n'est jamais parfait par nature). Si le

bruit de mesure des capteurs est en général gaussien, ce qui facilite le calcul de R, le "process noise" est lui souvent beaucoup plus complexe. De nombreuses applications utilisant le filtre de Kalman font alors appel à une méthode empirique en prenant Q de la forme $q \times I_{10}$. Différentes valeurs de q sont essayées sur une trajectoire type, et celle minimisant l'erreur quadratique entre l'estimation du filtre de Kalman et la valeur réelle est retenue. Afin de disposer d'une valeur de départ, nous avons réalisé des simulations avec un process noise aléatoire, en mesurant l'erreur quadratique cumulée en fonction de la valeur de Q. Nous avons pu obtenir une valeur de q de l'ordre de 10×10^{-5}, avec un minima en 2×10^{-5}, ce qui est visible en figure 22. Il ne s'agit cependant que d'une valeur de départ que l'on devra ajuster lors des tests complets de la fusée.

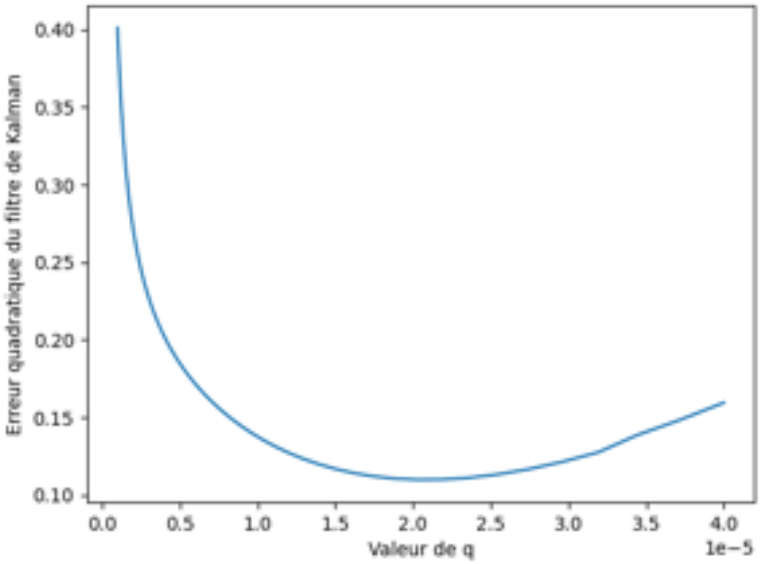
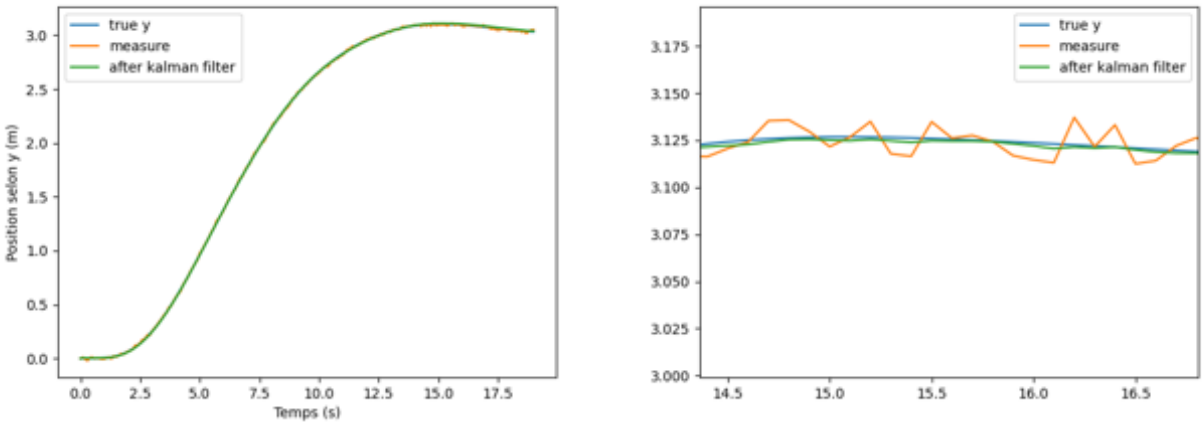


Figure 22: Détermination de la valeur optimale de q

Une fois les différents paramètres du filtre de Kalman déterminée nous avons pu le tester en le soumettant à du bruit gaussien. Le résultat selon l'axe y est visible en figure 23. On observe que le filtre de Kalman parvient bien à éliminer la grande majorité du bruit de mesure en lissant la courbe, la valeur estimée de y se confond alors presque avec le y réel.

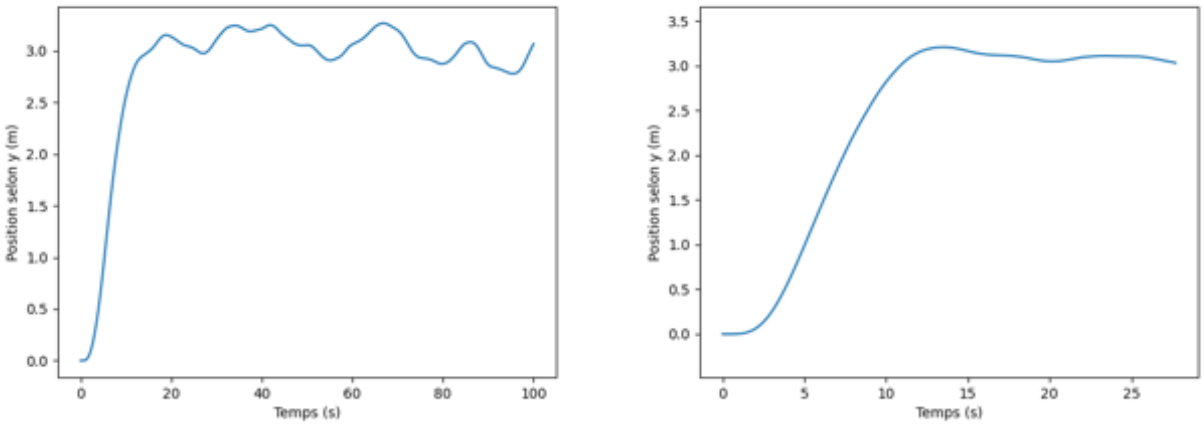


(a) Position selon l'axe y

(b) Zoom sur une portion de la courbe

Figure 23: Essai du filtre de Kalman

On peut alors se demander ce qu'il se passerait si l'on n'utilisait pas de filtre de Kalman, c'est à dire si la position mesurée par l'IMU était directement envoyée en entrée des algorithmes de contrôle. Il se trouve qu'après essai, cela ne modifie quasiment pas la trajectoire de la fusée pour un bruit borné par un centimètre, comme c'est le cas pour les valeurs mesurées par notre IMU. C'est une bonne nouvelle car cela montre la robustesse de notre PID et de notre LQR. On a cependant essayé d'augmenter artificiellement le bruit de mesure jusqu'à percevoir une nette différence lors de l'utilisation du filtre de Kalman. Ainsi pour un bruit borné par 10 centimètres, on obtient les courbes visibles en figure 24. On observe que sans filtre de Kalman, le LQR à du mal à faire face à un bruit aussi important et ne parvient pas à converger, oscillant autour de la valeur consigne même au bout d'une centaine de secondes. En présence du filtre, le comportement est bien meilleur, avec une convergence en 28 secondes environ, ce qui montre tout l'intérêt du filtre de Kalman pour le contrôle de notre fusée.



(a) Sans filtre de Kalman

(b) Avec filtre de Kalman

Figure 24: Comparaison sans/avec filtre de Kalman avec un bruit de mesure important

3.3 Pistes futures

L'utilisation du nouveau banc de test que nous avons conçu va non seulement permettre d'affiner notre connaissance de la dynamique de la fusée, grâce à des mesures plus précises selon tous les axes, mais aussi et surtout permettre de faire des test captifs de la fusée et de ses algorithmes de contrôle. Cela permettra notamment d'affiner le choix des différents paramètres de ces algorithmes.

À terme, lorsque la fusée sera fonctionnelle, il pourrait être envisageable d'essayer d'autres algorithmes de contrôle de la fusée et d'effectuer une étude comparative. Les options possibles sont nombreuses (PID, LQR, LQG, L1, H1, SMC, FBL, Backstepping, Fuzzylogic, Neural-networks, Genetic) [14]. Chacune dispose d'avantages et d'inconvénients (robustesse, facilité d'implémentation, adaptabilité, optimalité, précision...). Ainsi, même une fusée parfaitement fonctionnelle pourrait servir de sujet d'étude sur une année complète ou plus.

4 Communication et site web

Lorsque nous avons hérité du projet, nous avons fait face à un important temps de prise en main des différentes problématiques sur lesquels nous devions travailler. Le projet nécessite en effet comme l'ensemble des PSC du Centre Spatial une candidature tôt dans l'année. Il est assez difficile de s'informer en profondeur sur BackOnEarth avant cette candidature. De plus, une fois qu'il nous a été attribué, le seul document à notre disposition était le rapport de l'année précédente. Celui-ci, en tant que rapport final, est cependant plus proche d'un compte rendu exhaustif du travail effectué durant l'année qu'une présentation de BackOnEarth simple et facile à appréhender.

Par ailleurs, si BackOnEarth comptait plusieurs soutiens en dehors de l'École (VectorNav, MBDA) à son lancement, ceux-ci se sont perdus en 2020 avec la crise sanitaire et nous avons eu du mal à en nouer de nouveau malgré des tentatives avec l'ONERA notamment pour la réalisation du banc de test. Encore une fois le besoin de pouvoir présenter et vulgariser le projet s'est fait sentir, pour faciliter la prise de contact avec d'autres partenaires.

Ainsi, à la lumière de ces événements et dans le cadre du modal Web d'un des membres de l'équipe, nous avons décidé de créer un site web pour BackOnEarth. Les objectifs sont multiples :

- Communiquer sur le projet (4.1) ;
- Faciliter la collaboration avec des acteurs extérieurs (4.1) ;
- Permettre aux candidats des futures années de se présenter (4.2) ;
- Créer une continuité en informant sur les nouvelles avancées (4.3).

Le dernier point est notamment important car le projet BackOnEarth est appelé à durer. Comme de nombreux projets du Centre Spatial il est probable que ses développements se conduisent encore sur plusieurs années. Les promotions ne restant sur le campus que deux ans, il nous semblait donc important de disposer d'un outil capable de maintenir une continuité au sein du projet. Nous avons veillé à ce que la mise à jour du site puisse être faite même si les prochaines équipes BackOnEarth ne disposent pas de membre ayant des compétences de développement web. Le site est en anglais, pour faciliter le travail avec d'éventuels partenaires non-francophones comme VectorNav, ou si un étudiant non-francophone souhaitait se joindre au projet à l'avenir. Le site est disponible à l'adresse suivante : https://backonearth.binets.fr.

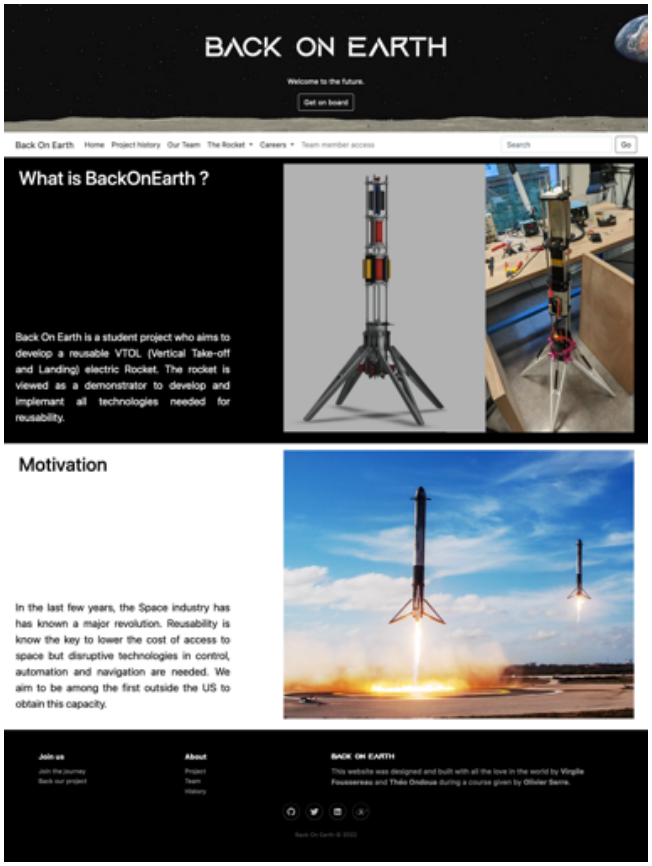


Figure 25: Page d'accueil du site web du projet BackOnEarth

4.1 Visiteur du site

Tout visiteur du site peut accéder à la présentation du projet, qui permet d'avoir une bonne compréhension de la fusée et de ses composants sans rentrer dans des détails trop techniques. Il peut également consulter la page de l'équipe qui présente les membres du projet avec leur rôle au sein de BackOnEarth, ce qui peut par exemple faciliter la communication inter-promotion. Cette page est automatiquement mise à jour dès qu'une nouvelle équipe s'inscrit sur le site, sans code supplémentaire nécessaire. Une fonction recherche permet de chercher un contenu précis ("Centrale inertielle" par exemple) sur le site, ce qui sera notamment utile si le contenu se densifie au cours des années. Enfin un espace partenaire permet aux acteurs extérieurs (publics ou entreprises) de nous contacter pour un éventuel partenariat.

4.2 Candidat

Le PSC BackOnEarth faisant l'objet d'une sélection en début d'année, nous avons créé un espace candidat sur le site pour faciliter la tâche du Centre Spatial. Elle permet notamment l'envoi de la motivation du candidat. Le tout est envoyé de façon sécurisée sur le serveur et est alors disponible pour être consulté. Un mail automatique signale la candidature. Le site étant

hébergé par l'école, nous avons travaillé avec la DSI pour en assurer la sécurité via plusieurs audits de sécurité et l'installation de captchas.

4.3 Membre de l'équipe BackOnEarth

Chaque membre du projet peut créer son compte sur le site du projet, ce qui nécessite le mot de passe du projet. Une fois inscrit, il a accès à plusieurs fonctionnalités dont la principale est l'ajout de contenu. En effet, le projet étant amené à évoluer au fur et à mesure des années, il est important de pouvoir régulièrement ajouter du contenu sur le site, sans forcément disposer de compétences en développement web. L'ajout de contenu se fait de façon très simple via un formulaire : un titre, un court paragraphe de texte et une image. Une fois ces informations rentrées, le site automatise entièrement le formatage (police d'écriture, taille de l'image) et notamment l'alternance de noir et blanc pour chaque bloc de contenu sur une page.

Bien sûr, en cas d'erreur, il est possible pour un membre du projet de supprimer un contenu. Pour cela il suffit de fournir le titre du contenu à supprimer (de façon exacte, afin d'éviter les erreurs). Même en supprimant un contenu en plein milieu de page, l'alternance de noir et de blanc sera conservée pour que la présentation reste propre.

Conclusion

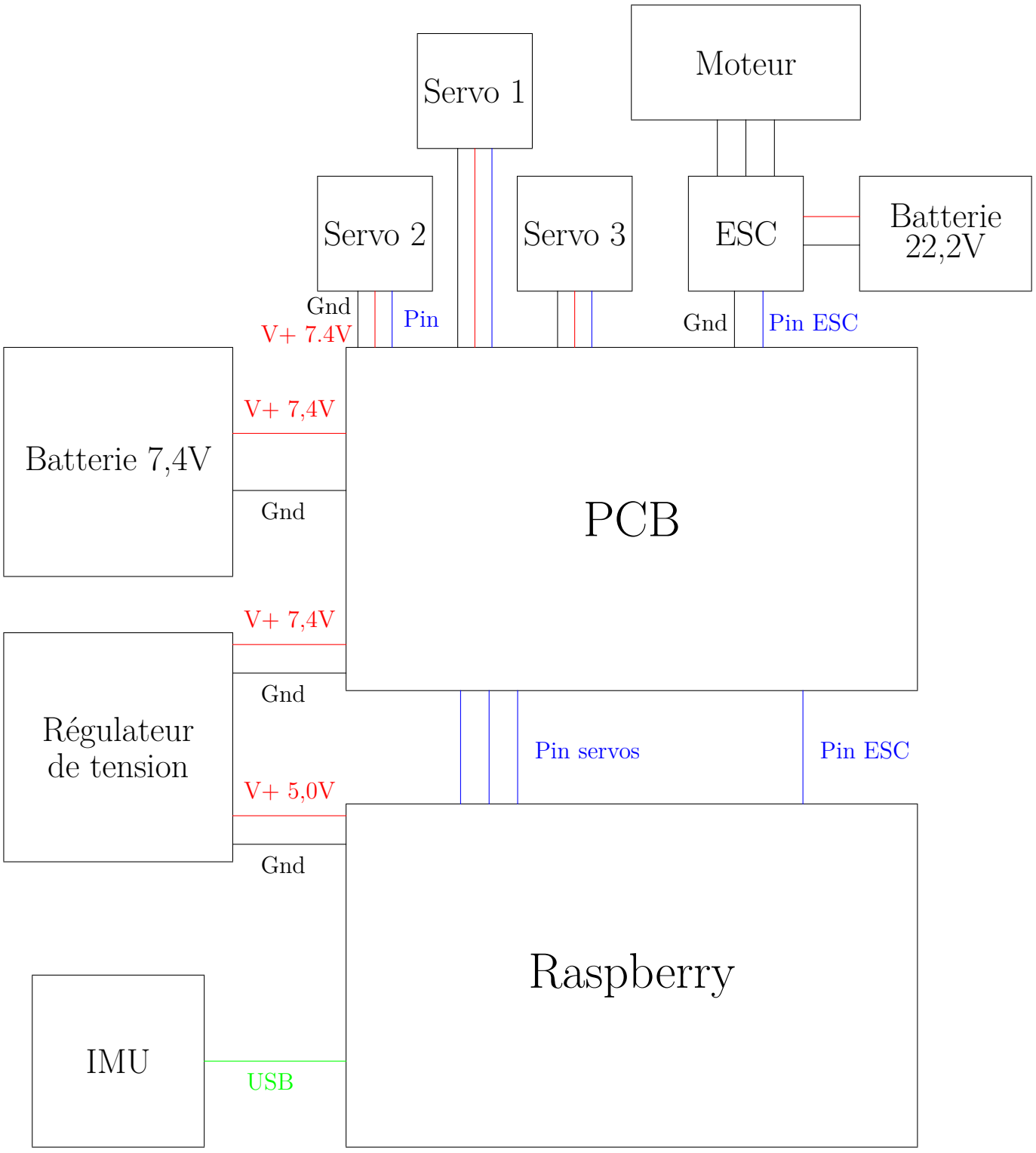
Avant d'achever notre travail en récapitulant l'état actuel de la fusée ainsi que les avancées techniques et scientifiques que nous avons apportées à ce projet, il est important de revenir sur les objectifs que nous nous étions fixés dès le début de celui-ci.

En début d'année, nous avions trois objectifs principaux dont le premier était d'achever le design et l'assemblage de l'architecture de la fusée et de valider par une étude quantitative l'action du TVC afin de l'inclure adéquatement dans l'algorithme de contrôle. Notre deuxième objectif était de concevoir et de programmer l'algorithme de contrôle afin de l'intégrer dans la mémoire de la Raspberry qui commande la fusée. Enfin, nous voulions valider la construction de la fusée, tant mécanique qu'informatique, en parvenant à la faire voler de façon prévisible et contrôlée.

À ce stade, le premier objectif est à moitié rempli puisque l'architecture et le design de la fusée sont achevés mais que l'étude quantitative n'a pas encore donné de résultats d'une précision acceptable. Le deuxième objectif est, quant à lui, atteint car le programme de contrôle de la fusée est écrit, en intégrant les données de position de la fusée données par l'IMU et en commandant les ailettes du TVC et la puissance moteur pour rectifier selon la trajectoire prévue. Enfin, le troisième objectif n'est pas encore atteint. Cependant, le banc d'expérimentation complexe à 3 degrés de liberté que nous avons construit ces derniers mois est enfin fonctionnel. Il permettra de fournir, très vite nous l'espérons, les données d'action des ailettes sur la correction de la position et ensuite de permettre les premiers vols en semi-liberté.

Ainsi, dans l'hypothèse souhaitable où ce projet *Back on Earth* serait poursuivi par un groupe l'an prochain, nous léguons à ce dernier l'ensemble des moyens techniques et informatiques requis pour obtenir les données quantitatives manquantes pour pouvoir commencer la phase d'essais en vols.

A Schéma électrique de la fusée



B Calculs de dynamique

On prend le repère $R = (O, \mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z})$ lié au laboratoire et le repère $R' = (O, \mathbf{e'_x}, \mathbf{e'_y}, \mathbf{e'_z})$ construit selon le changement de base suivant :

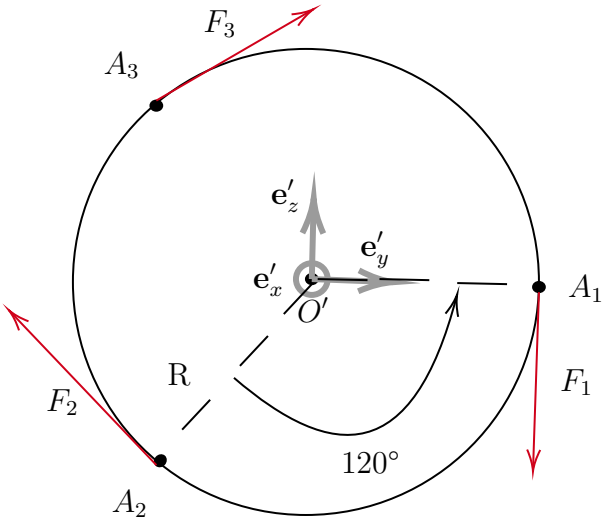


Figure 26: Schéma du TVC vu d'en-dessous

On note A_i le point d'attache de chaque ailette et O' le centre du support des ailettes, à la base de la fusée. On note G le centre de gravité de la fusée avec $\mathbf{O'G} = L\mathbf{e_x}$ et $\mathbf{O'A_i} = R\mathbf{e_y}$.

D'après la relation de Varignon :

$$\begin{aligned}\mathbf{M_G}(\mathbf{F_1}) &= \mathbf{MA_1}(\mathbf{F_1}) + \mathbf{GA_1} \times \mathbf{F_1} \\ &= \mathbf{F_1} \times (L\mathbf{e_x} - R\mathbf{e_y}) \\ &= \begin{bmatrix} -F_1 R \\ -F_1 L \\ 0 \end{bmatrix}_{R'}\end{aligned}$$

On trouve de même :

$$\mathbf{M_G}(\mathbf{F_2}) = \begin{bmatrix} -F_2 R \\ \frac{1}{2} F_2 L \\ -\frac{\sqrt{3}}{2} F_2 L \end{bmatrix}_{R'}$$

et

$$\mathbf{M_G}(\mathbf{F_3}) = \begin{bmatrix} -F_3 R \\ \frac{1}{2} F_3 L \\ \frac{\sqrt{3}}{2} F_3 L \end{bmatrix}_{R'}$$

Ainsi, on a

$$\mathbf{M_G(F_{TOT}}) = \begin{bmatrix} -(F_1+F_2+F_3)R \\ (\frac{F_2+F_3}{2} - F_1)L \\ \frac{\sqrt{3}}{2}(F_3-F_2)L \end{bmatrix}_{R'} \simeq \begin{bmatrix} -(F_1+F_2+F_3)R \\ (\frac{F_2+F_3}{2} - F_1)L \\ \frac{\sqrt{3}}{2}(F_3-F_2)L \end{bmatrix}_{R}$$

au premier ordre.

On peut donc appliquer le théorème du moment cinétique, on trouve alors :

$$\ddot{\phi} = \frac{F_1+F_2+F_3}{J_x} R$$

$$\ddot{\theta} = (F_1 - \frac{F_2+F_3}{2}) \frac{L}{J_y}$$

$$\ddot{\psi} = \frac{\sqrt{3}}{2}(F_2 - F_3) \frac{L}{J_z} \tag{1}$$

Synthèse et passage sous forme matricielle

On définit le vecteur $W = \begin{bmatrix} y \\ \dot{y} \\ z \\ \dot{z} \\ \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix}$ et on modélise le comportement de notre système par l'équation matricielle $\dot{W} = AW + BU$, où on définit :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{F}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{F}{M} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

et

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}A}{2m} & -\frac{\sqrt{3}A}{2m} \\ 0 & 0 & 0 \\ -\frac{A}{m} & \frac{A}{2m} & \frac{A}{2m} \\ \frac{AR}{J} & \frac{AR}{J} & \frac{AR}{J} \\ 0 & 0 & 0 \\ \frac{L}{J} & -\frac{L}{2J} & \frac{L}{2J} \\ 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}L}{2J} & -\frac{\sqrt{3}L}{2J} \end{bmatrix}$$

C Code Python

```python
import numpy as np
from math import sqrt
import scipy.linalg as linalg
from simple_pid import PID
import time

#### Variables

max_fin_angle = 45 # degree
noiseAltitudeLevel=0.01
noiseAltitudeVelocityLevel=0.01
noisePositionLevel=0.01
noisePositionVelocityLevel=0.01
noiseAngleLevel=0.001
noiseAngleVelocityLevel=0.001
M  = 4
g  = 9.81
Poids  = M*g
Fe = M*g
a  = 0.1*0.25
L  = 0.3
J  = 0.4
d  = 0.05
dt = 0.1
Q_Kf=2e-5*np.eye(10)
Q_altitude=np.eye(2)

B = dt*np.array([  [0, 0, 0],
                   [0, sqrt(3)*a/(2*M), -sqrt(3)*a/(2*M)],
                   [0, 0, 0],
                   [-a/M, a/(2*M), a/(2*M)],
                   [0, 0, 0],
                   [a*d/J, a*d/J, a*d/J],
                   [0, 0, 0],
                   [L/J, -L/(2*J), -L/(2*J)],
                   [0, 0, 0],
                   [0, sqrt(3)*L/(2*J), -sqrt(3)*L/(2*J)]])

R = 1000*np.identity(3) # Penalty for angular velocity effort
Q = 0.1*np.identity(10)
Q[4,4]=1000
Q[6,6]=1000
Q[8,8]=1000

def getA(F):
    dA = np.array([ [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, F/M, 0],
                    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, -F/M, 0, 0, 0],
                    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
    A = np.eye(10) + dt*dA
    return A

A=getA(Fe)
KalmanFilterOn=True
KalmanFilterAltitudeOn=True
```

```
#### Initialization

vec_altitude_estimate=np.array([0,0])
vec_altitude_measure=np.array([0,0])
x_estimate = np.array([0,0,0,0,0,0,0,0,0,0])
x_measure=np.array([0,0,0,0,0,0,0,0,0,0])
x_final = np.array([3,0,3,0,0,0,0,0,0,0])
u=np.array([0,0,0])
cov=0.0001*np.identity(10)
Pk=0.0001*np.identity(2)
pid = PID(0.07, 0.0001,1, setpoint=3) #setpoint is the objective, here an altitude of 3 meters
# sans dépassement :   pid = PID(0.07, 0.0001,1, setpoint=3)
# rapide :     pid = PID(0.2, 0.0001,1.25, setpoint=3)
pid.sample_time=0


#### Kalman Filter

def KalmanFilterUpdate(A, B, X, u, P, z, q):

    xf, Pf =KalmanFilterPredict(A, B, X, u, P, q)

    #Computation of the Observation Matrix
    H=np.eye(10)

    #Computation of Measurement Uncertainty
    R=np.eye(10)
    for i in range(0,4,2):
        R[i,i]=pow(noisePositionLevel,2)
    for i in range(1,4,2):
        R[i,i]=pow(noisePositionVelocityLevel,2)
    for i in range(4,10,2):
        R[i,i]=pow(noiseAngleLevel,2)
    for i in range(5,10,2):
        R[i,i]=pow(noiseAngleVelocityLevel,2)


    #Computation of the Kalman gain
    Ht=H.transpose()
    K=Pf@Ht@np.linalg.inv(H@Pf@Ht+R)

    #State update
    x = xf+K@(z-H@xf)

    #Covariance update
    IKH=(np.eye(10)-K@H)
    IKHt=IKH.transpose()
    Pnew=IKH@Pf@IKHt@+K@R@(K.transpose())

    return x,Pnew

def KalmanFilterPredict(A, B, X, u, P, q):
    xf=(A @ X) + (B @ u)

    Pf=A@P@(A.transpose())+q

    return xf, Pf
```

```python
#Kalman Filter for altitude
def KalmanFilterAltitude(vecAltitude, Fe, P, vec_measure, q):

    vecAltitudef, Pf = KalmanFilterPredictAltitude(vecAltitude, Fe, P, q)

    #Computation of the Observation Matrix
    H=np.eye(2)

    #Computation of Measurement Uncertainty
    R=np.eye(2)
    R[0,0]=pow(noiseAltitudeLevel,2)
    R[1,1]=pow(noiseAltitudeVelocityLevel,2)


    #Computation of the Kalman gain
    Ht=H.transpose()
    K=Pf@Ht@np.linalg.inv(H@Pf@Ht+R)

    #State update
    vecAltitude = vecAltitudef+K@(vec_measure-H@vecAltitudef)

    #Covariance update
    IKH=(np.eye(2)-K@H)
    IKHt=IKH.transpose()
    Pnew=IKH@Pf@IKHt@+K@R@(K.transpose())+q

    return vecAltitude, Pnew

def KalmanFilterPredictAltitude(vecAltitude, Fe, P, q):
    vecAltitudef=A_altitude@vecAltitude + B_altitude*(Fe-Poids)/M
    P=A_altitude@P@(A_altitude.transpose())+q
    return vecAltitudef, P


#### Step

def step(xk, cov, vec_altitude_estimate, Pk, vec_altitude_measure, x_measure, x_final, u, Fe):

    A=getA(Fe)

    #Kalman Filter on measures
    if (KalmanFilterOn):
        xk, cov = KalmanFilterUpdate(A, B, xk, u, cov, x, Q_Kf)
    else:
        xk=x_measure
    if (KalmanFilterAltitudeOn):
        vec_altitude_estimate, Pk = KalmanFilterAltitude(vec_altitude_estimate, Fe, Pk, vec_altitude_measure, Q_altitude)
    else:
        vec_altitude_estimate=vec_altitude_measure

    #PID for altitude control
    Fe=pid(vec_altitude_estimate[0],dt)+Poids

    #LQR for horizontal and angular control
    A=getA(Fe)
    P = linalg.solve_discrete_are(A, B, Q, R, e=None, s=None, balanced=True)
    F_lqr = np.linalg.pinv(R+B.T@P@B) @ B.T @ P @ A  # F deja pris...
    u = -F_lqr @ (xk - x_final)

    return u, Fe, xk, cov, vec_altitude_estimate, Pk
```

```python
### Simulation

x=np.array([0,0,0,0,0,0,0,0,0,0,0,0])
vecAltitude=np.array([0,0])
i=0
error = np.linalg.norm(x - x_final)
X=[x[0]]
T=[0]
while (i<1000):
    print(f't = {dt*i}s')
    T.append(T[-1]+dt)
    u, Fe, x_estimate, cov, vec_altitude_estimate, Pk = step(x_estimate, cov, vec_altitude_estimate, Pk, vec_altitude_measure, x_measure, x_final, u, Fe)

    vecAltitude=A_altitude@vecAltitude + B_altitude*(Fe-Poids)/M
    x=(A @ x) + (B @ u)
    X.append(x[0])

    noisePosition=np.random.normal(0,noisePositionLevel,4)
    noiseAngle=np.random.normal(0,noiseAngleLevel,6)
    noise = np.concatenate([noisePosition,noiseAngle])
    x_measure=x+noise
    vec_altitude_measure=vecAltitude

    error = np.linalg.norm(x - x_final)
    i+=1
    if error < 0.1:
        print("\nGoal Has Been Reached Successfully!")
        break
plt.figure()
plt.plot(T,X, label='true y')
plt.show()
```

Figure 27: Code Python du système de contrôle

Bibliographie

[1] *Understanding PWM*. Aug. 2009. URL: https://ebldc.com/?p=48.

[2] *Documentation VectorNav. AVN-300 User Manual*.

[3] *Documentation VectorNav. Quick Start Guide*.

[4] NovAtel Inc. *An Introduction to GNSS*. 2015. Chap. 6. URL: https://novatel.com/an-introduction-to-gnss.

[5] Mark Lundberg. *Proportional–integral–derivative controller implementation in Python*. 2021. URL: https://github.com/m-lundberg/simple-pid.

[6] Rudolph Emil Kalman. "When Is a Linear Control System Optimal?" In: *Journal of Basic Engineering* 86.1 (Mar. 1964), pp. 51–60. ISSN: 0021-9223. DOI: 10.1115/1.3653115.

[7] Russ Tedrake. *Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2022. Chap. 8. URL: http://underactuated.mit.edu.

[8] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82 (1960), p. 35. DOI: 10.1115/1.3662552.

[9] Kenshi Saho and Masao Masugi. "Performance analysis of $\alpha - \beta$ tracking filters using position and velocity measurements". In: *EURASIP Journal on Advances in Signal Processing* 2015 (Dec. 2015). DOI: 10.1186/s13634-015-0220-3.

[10] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. 1995.

[11] Becker, Alex. *Kalman Filter Project*. 2022. URL: https://www.kalmanfilter.net.

[12] Hari Bansal, Rajamayyoor Sharma, and Shreeraman Ponpathirkoottam. "PID Controller Tuning Techniques: A Review". In: 2 (Nov. 2012), pp. 168–176.

[13] Qing-Guo Wang et al. "PID tuning for improved performance". In: *IEEE Transactions on Control Systems Technology* 7.4 (1999), pp. 457–465. DOI: 10.1109/87.772161.

[14] Samuel John and Andrew Zulu. "A Review of Control Algorithms for Autonomous Quadrotors". In: *Open Journal of Applied Sciences* 4.14 (Dec. 2014), pp. 547–556. DOI: 10.4236/ojapps.2014.414053.