Projet Skyscanner

Optimisation d'un drone d'observation des nuages

Lauren Dargent, Raphaël Julé, Vincent Lefèvre et Pierre Mermet-Bijon

ISAE - Supaéro

Projet Skyscanner

Rapport final

Optimisation d'un drone d'observation des nuages

Auteurs :
Lauren Dargent
Raphaël Julé
Vincent Lefèvre
Pierre Mermet-Bijon

Encadrants :
Emmanuel Bénard
El Khedim Bouhoubeiny
Franck Gormand

25 mars 2016

Résumé

Le projet Skyscanner est issu du besoin du CNRM pour un drone d'observation des nuages en vue de valider et d'affiner leurs modèles aérologiques de formation des orages pour Météo-France. Les spécifications du CNRM ont menés à considérer un mini-drone de longue endurance : un concept aujourd'hui encore peu avancé du fait de l'apparente incompatibilité entre faible taille et longue endurance relevée dans de nombreux travaux jusqu'ici. Cependant, certaines idées récentes basées notamment sur une utilisation astucieuse de l'aérologie des zones orageuses nous autorisent à imaginer des drones de faible taille et à grande autonomie qui pourraient satisfaire au besoin du CNRM. C'est donc dans cette optique que le CNRM a rassemblé plusieurs acteurs (LAAS, ENAC, Supaéro) autour du projet Skyscanner pour développer un tel drone.

Au sein de ce projet, le travail de notre équipe (Supaéro) a consisté à développer un outil de développement de drone. La portée de cet outil dépasse le cadre du projet conformément à la volonté d'Emmanuel Bénard de développer un outil qui soit le plus général possible, et réexploitable pour d'autres projets. A cette fin, nous avons développé 2 outils numériques complémentaires qui seront présentés dans ce rapport : le premier est un programme Excel permettant de convertir des spécifications mission en spécifications drone, il évalue les performances minimales requises à un drone pour réaliser une mission donnée. Le second, est un programme d'optimisation multidisciplinaire qui optimise la configuration d'un drone pour maximiser un critères de performance tout en respectant un ensemble de contraintes liées à la mission du drone.

Table des matières

Remerciements

Nous tenons à remercier nos tuteurs de projet M. Emmanuel Benard, M. El Khedim Bouhoubeiny pour avoir proposé ce projet et nous avoir soutenus pendant son déroulement. Ce fut pour nous une expérience très enrichissante et très plaisante de travailler sur le projet Skyscanner sous leur supervision.

Nous remercions aussi M. Franck Gormand et M. Rob Vingerhoeds pour leurs conseils avisés sur la conduite du projet.

Enfin, nous remercions M. Greg Roberts du *CNRM* et M. Simon Lacroix de l'*ONERA* pour leurs explications sur les missions d'observation de nuages.

Chapitre 1

Introduction

Ce document est un rapport de projet effectué en troisième année d'école d'ingénieur à l'*ISAE-SUPAERO*. Le projet Skyscanner est un projet de recherche scientifique visant à développer une flotte de drone et à la faire évoluer autour de nuages. Ces drones ont pour mission de suivre l'évolution des nuages et d'effectuer des mesures météorologiques autour de ces derniers. Pour accomplir ces missions, les drones doivent voler durant plusieurs heures. Cette endurance est obtenue par l'optimisation fine de la géométrie des appareils. Notre groupe projet s'inscrit dans la démarche de conception des drones, encadré par MM. Bénard, Bouhoubeiny et Gormand.

Ce rapport récapitule les quatre mois de travail sur ce projet. La première partie de ce rapport est consacrée à la définition du cahier des charges de la mission du drone en coopération avec le CNRM. La seconde partie détaille le programme développé : dans un premier temps, les blocs mono-disciplinaires du code seront explicités, et dans un second temps, c'est la logique d'optimisation qui sera présentée. Enfin, la troisième partie du document est consacrée à la gestion du projet.

1.1 Présentation du contexte général

Le projet *SKYSCANNER* consiste à définir des solutions et stratégies de déploiement permettant de mettre en œuvre une flotte de micro-drones capable de prendre des mesures autour et à l'intérieur de nuages. Le projet est constitué de trois acteurs : le CNRM (Centre National de Recherche Météorologique), le LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) et l'ISAE (Institut Supérieur de l'Aéronautique et de l'Espace) avec l'ONERA (Office National d'Études et de Recherches Aérospatiales).

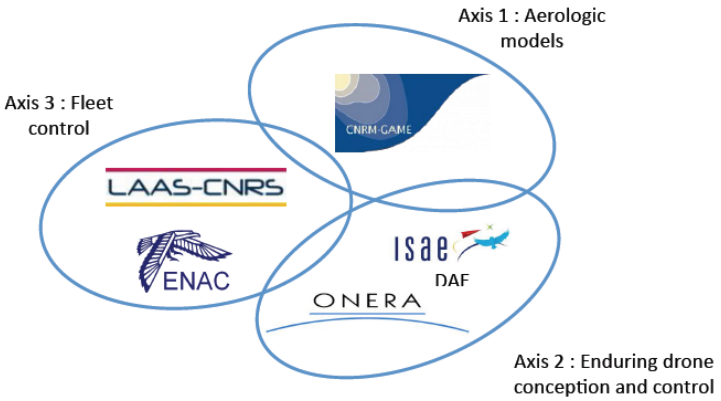


FIGURE 1.1 – Partenaires du projet *Skyscanner*

1.2 Notre cadre du projet

Notre équipe, constituée de 4 personnes : Lauren DARGENT, Raphaël JULÉ, Vincent LEFÈVRE et Pierre MERMET-BIJON est comprise au sein du groupe composé de l'ISAE et de l'ONERA (*Axis 2*) et chargé de la conception et le contrôle d'une drone à longue endurance. Elle est encadrée par les tuteurs Emmanuel BENARD, El Khedim BOUHOUBEINY et Franck GORMAND. Le travail de l'ISAE-*Skyscanner* consiste à développer un drone d'observation et d'analyse des nuages et un programme de design de drone. Plus précisément, notre travail s'est articulé autour de trois axes :

— **La définition de la mission avec le CNRM**
— **L'amélioration du programme d'optimisation**
— **La validation de notre programme**

Chapitre 2

Définition de la mission

2.1 Présentation de la mission définie avec le CNRM

Lorsque notre équipe est arrivée sur le projet, la mission que devaientt effectuer nos drones était très peu définie, et en conséquent il était difficile de concevoir les drones sans cahier des charges prédéfini. Aussi, notre première mission a consisté à définir plus précisément le cahier des charges de notre drone afin qu'il puisse mener à bien sa mission. Nous avons notamment dû définir clairement ce qu'est exactement que la mission moyenne d'un drone, ainsi que la variance associée à cette mission.
En collaboration avec M. Greg Roberts du Centre National de Recherche en Météorologie (CNRM), nous avons donc défini la mission que les drones du projet Skyscanner devait remplir.

L'objectif de ces drones est de prendre des mesures météorologiques dans et autour un cumulus en formation. Pour une mission, trois drones sont lancés.

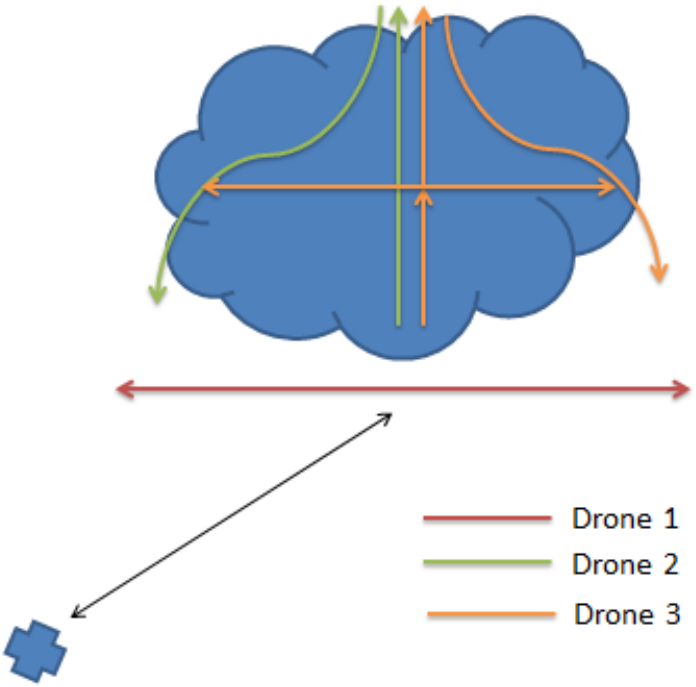


Figure 2.1 – Les 3 profils de mission

Les trois drones partent de la base pour rejoindre un point de référence situé à la base du nuage. A partir de ce point de référence, on se place dans le repère lié au nuage (et donc mobile par rapport au

sol, dépendamment du vent).
Le premier drone reste au-dessous du nuage, et parcourt ses diamètres. Il a pour but de prendre diverses mesures et de repérer les ascendances. Le deuxième drone monte en spirale au sommet du nuage, puis redescend sur les côtés tout en prenant des mesures. Enfin le troisième drone monte également au sommet du nuage, mais s'arrête à mi-hauteur pour mesurer l'extension latérale du cumulus. Il fait donc une portion de vol en palier. Ensuite, il monte au sommet du nuage et redescend sur le côté de la même manière que le deuxième drone. Enfin, une fois la mission finie, les drones rejoignent la base située au sol.

Les drones sont amenés à parcourir trois profils de mission différents, nous décidons donc dans un premier temps de designer un drone pour chacune de ces missions.
Outre la définition des profils de mission, le travail avec le CNRM nous a permis d'élaborer un cahier des charges concernant les performances attendues pour les drones. Ce cahier des charges est représenté dans le tableau suivant :

Critère	Valeur
Autonomie (h)	1
Charge utile (kg)	0,5
Vitesse minimale (m/s)	15
Vitesse de montée (m/s)	2
Rayon de virage (m)	100
Altitude maximale (haut du nuage) (m)	3000
Altitude de la base du nuage (m)	800
Envergure maximale (m)	1,5
Distance maximale à la base (km)	6

FIGURE 2.2 – Cahier des charges : performances du drone

De plus, nous avons établi un cahier des charges plus spécifique à l'opération des drones :

Critère	Commentaires
Structure	En polystyrène
Accès aux capteurs	Doit être aisée
Propulsion	Hélice arrière/sous les ailes
Décollage	Sur rampe, doit nécessiter une accroche sous le fuselage
Atterrissage	Dans un filet
Batterie	Electrique

FIGURE 2.3 – Cahier des charges : opération

Enfin, nous avons également déterminé quels seraient les capteurs à utiliser pour cette mission. Ces capteurs embarqués doivent à la fois stocker les informations sur une carte SD (échantillonnage à 30 Hz) à bord du drone, et les transmettre au sol (échantillonnage à 1 Hz). Les capteurs embarqués et leurs caractéristiques sont décrites dans le tableau suivant :

Capteur	Caractéristiques	Rôle	Prise de mesures
Nuage	5x8cm sur le côté du fuselage	Mesure l'extension du nuage	--
Vecteur de vent 3D	Tige de 15cm sur le nez du drone La pluie peut boucher les orifices Ne doit pas heurter le sol.	Reconstitue le vecteur vitesse du vent en 3D	Altitude constante, drone en ligne droite
Capteur de flux solaire	Dans le nez du drone Ne doit pas toucher le sol	Mesurer le flux solaire	Altitude constante, drone en ligne droite
Pitot	Placé sur les ailes si pas de vecteur vent 3D	Mesure la pression	--
Caméra FPV (1st Person View)	Placée sous une des ailes	Permet de filmer l'environnement sous le drone	--

FIGURE 2.4 – Cahier des charges : capteurs du drone

2.2 Formulation de la mission pour le code

Maintenant que nous avons défini les profils de mission et les performances attendues avec le CNRM,nous devons être en mesure d'adapter ces informations pour les prendre en compte dans notre programme d'optimisation.
Nous allons donc créer un programme de conception de drone optimisés pour un certain profil de mission et nous livrerons trois drones, chacun optimisé pour le profil de mission défini avec le CNRM dans le paragraphe précédent.
Nous avons donc créé un fichier *Excel* $specifications\_mission.xslx$ qui permet de prendre en entrée les différents paramètres de mission, et en sortie de renvoyer plusieurs paramètres de cas de vol. Ces paramètres sont ensuite mis en entrée de notre programme d'optimisation, et pour chaque jeu de paramètres on conçoit un drone optimisé pour un des trois profils types de mission.

1er onglet : Paramètres

La première page de ce classeur *Excel* permet de regrouper tous les paramètres d'entrée de notre mission. Ceux-ci sont regroupés en plusieurs catégories, avec pour chacune la valeur retenue pour la mission (case orange, modifiable) et les valeurs minimales et maximales imposées par le CNRM, quand elles existent (cases gris clair, à droite des cases oranges).

Conditions atmosphériques Ces paramètres constituent les caractéristiques atmosphériques de la mission, comme la température au niveau du sol, l'altitude au niveau de la base, le vent (nous avons

considéré un vent constant et uniforme à tous les niveaux d'altitude), et les caractéristiques du nuage (position, extension, hauteur, ascendances éventuelles, etc).

Caractéristiques du drone Ces paramètres regroupent les caractéristiques du drone sur sa mission : vitesse horizontale et verticale, rayon de virage, masse et surface alaire. Le drone optimisé sera calculé à partir de ces paramètres-ci, mais ne convergera pas forcément vers ce jeu de paramètres : il est juste indicatif (mis à part certains paramètres,comme la masse maximale, qui sont de vraies contraintes sur notre drone et donc entrées en dur dans notre optimiseur comme des contraintes à respecter).

Caractéristiques de la mission Ces paramètres regroupent les caractéristiques nominales pour la mission que sont le temps de mission sur zone et la distance du nuage à la base en début de mission.

Cas de vol Nous parlerons juste en dessous des différents cas de vol que nous avons implémentés ; cette partie permet de régler le pourcentage de virages et de lignes droites pour chacun de ces cas de vol.

Nous noterons qu'il existe également des cases gris foncé avec les constantes utilisées pour les calculs. Ces cases ne doivent en aucun cas être modifiées.
La figure ci-dessous montre les différents paramètres à remplir en entrée de notre fichier *Excel* :

Conditions atmosphériques			
Température au sol(°C)	15	0	-
Altitude au sol/niveau de la mer (m)	100	0	-
Vitesse du vent (m/s)	5	0	15
Orientation du vecteur vitesse vent (°)	0	0	360
Base du nuage (m)	800	800	-
Hauteur du nuage (m)	2000	0	3000
Rayon du nuage (m)	600	-	6000
Vitesse verticale de l'ascendance (m/s)	0	-	-
Caractéristiques du drone			
Vitesse horizontale du drone (m/s)	20	5	35
Vitesse verticale du drone (m/s)	3	1	-
Vitesse de descente le long du nuage (m/s)	3	-	-
Rayon de virage du drone (m)	100	100	-
Longueur lignes droites (m)	2000	-	6000
Masse du drone (kg)	2,5	2	2,5
Surface alaire (m²)	1	-	-
Caractéristiques de la mission			
Temps de mesure (min)	30	30	-
Distance du nuage à la base en début de mission(m)	6000	0	6000

Cas de vol (à modifier)		
Rejoindre nuage	Virage (%)	10%
	Lignes droite	90%
Retour base	Virage (%)	10%
	Lignes droite	90%
Ascendances	Virage (%)	100%
	Lignes droite	0%
Inspection nuage	Virage (%)	50%
	Lignes droite	50%

Constantes	
gravitationn	9,81
gamma	1,4
R_air	287
densité au n	1,225

FIGURE 2.5 – onglet Parametres du livrable Excel

2eme onglet : Cas de vol

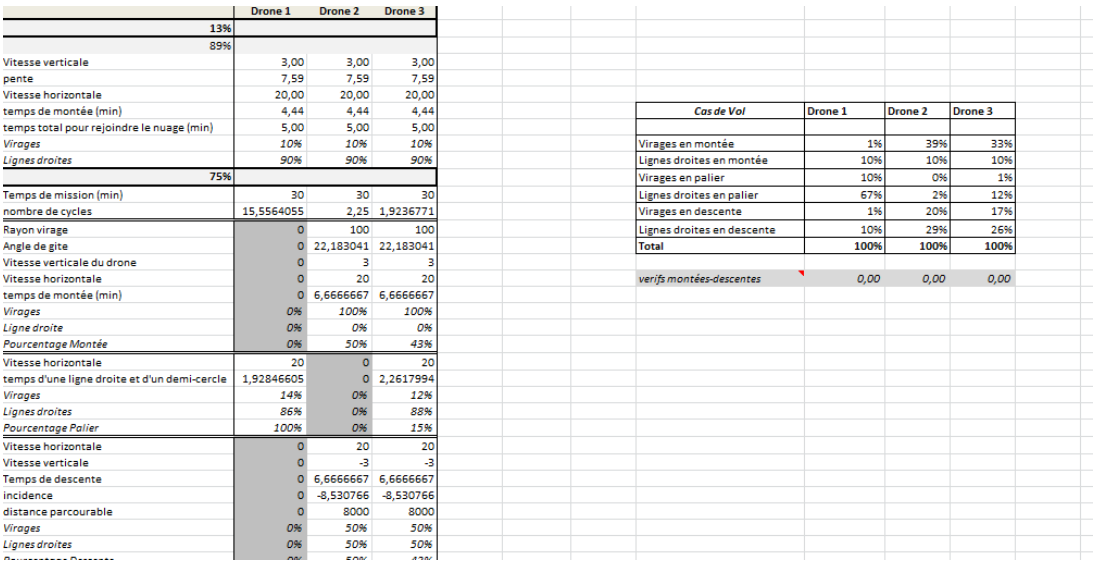
Notre programme est un logiciel d'optimisation qui cherche à minimiser un critère, à savoir dans notre cas : l'endurance de notre drone. Cette endurance est calculée à partir du profil de mission type de notre drone, et diffère selon le cas de vol : selon que le drone est en montée ou en palier, en virage ou en ligne droite, il n'aura pas de même C_d, ne consommera pas la même quantité d'énergie et donc n'aura pas la même endurance sur cette portion de la mission.
Nous avons donc défini six cas de vol ; chaque cas de vol correspond au pourcentage de temps que le drone aura passé dans cette configuration sur le temps total de mission. Nous avons donc :
$T_{cas\_de\_vol\_i} = \%_{cas\_de\_vol\_i} * T_{total\_mission}$
et la somme de tous les pourcentages de cas de vol doit être égale à 1.

Les cas de vol sont les suivants :

- Virages en montée
- Lignes droites en montée
- Virage en palier
- Ligne droite en palier
- Virage en descente
- Ligne droite en descente

Dans le deuxième onglet de notre fichier *Excel*, nous calculons ces pourcentages de cas de vol à partir des paramètres rentrés dans l'onglet précédent (Parametres). Pour cela, nous allons calculer pour chacun des trois drones leurs différentes vitesses, altitudes, distance parcourue, etc. sur plusieurs phases de la

mission et nous en déduisons le temps passé dans chaque cas de vol.
La figure suivante montre où récupérer les pourcentages pour chaque cas de vol :



	Drone 1	Drone 2	Drone 3
13%			
89%			
Vitesse verticale	3,00	3,00	3,00
pente	7,59	7,59	7,59
Vitesse horizontale	20,00	20,00	20,00
temps de montée (min)	4,44	4,44	4,44
temps total pour rejoindre le nuage (min)	5,00	5,00	5,00
Virages	*10%*	*10%*	*10%*
Lignes droites	*90%*	*90%*	*90%*
75%			
Temps de mission (min)	30	30	30
nombre de cycles	15,5564055	2,25	1,9236771
Rayon virage	0	100	100
Angle de gite	0	22,183041	22,183041
Vitesse verticale du drone	0	3	3
Vitesse horizontale	0	20	20
temps de montée (min)	0	6,6666667	6,6666667
Virages	*0%*	*100%*	*100%*
Ligne droite	*0%*	*0%*	*0%*
Pourcentage Montée	*0%*	*50%*	*43%*
Vitesse horizontale	20	0	20
temps d'une ligne droite et d'un demi-cercle	1,92846605	0	2,2617994
Virages	*14%*	*0%*	*12%*
Lignes droites	*86%*	*0%*	*88%*
Pourcentage Palier	*100%*	*0%*	*15%*
Vitesse horizontale	0	20	20
Vitesse verticale	0	-3	-3
Temps de descente	0	6,6666667	6,6666667
incidence	0	-8,530766	-8,530766
distance parcourable	0	8000	8000
Virages	*0%*	*50%*	*50%*
Lignes droites	*0%*	*50%*	*50%*

Cas de Vol	Drone 1	Drone 2	Drone 3
Virages en montée	1%	39%	33%
Lignes droites en montée	10%	10%	10%
Virages en palier	10%	0%	1%
Lignes droites en palier	67%	2%	12%
Virages en descente	1%	20%	17%
Lignes droites en descente	10%	29%	26%
Total	**100%**	**100%**	**100%**
verifs montées-descentes	*0,00*	*0,00*	*0,00*

FIGURE 2.6 – onglet Cas de vol du livrable Excel

3eme onglet : ParamAVL

Cet onglet calcule certains paramètres en fonction des conditions de vol (température, altitude) ou phases de vol (montée, descente, palier). L'utilisateur rentre :

- l'altitude du sol par rapport au niveau de la mer
- la température au sol
- la vitesse de croisière souhaitée
- le rayon de virage désiré
- les vitesses verticales voulues

et récupère en sortie plusieurs paramètres qui serviront dans un deuxième temps en tant qu'entrées à notre programme d'optimisation :

- densité de l'air aux différentes altitudes
- nombre de Mach
- Coefficient de portance en palier et en virage
- Incrément de poussée nécessaire pour monter/descendre
- Facteur de charge en virage

Conditions atmosphériques	SOL	base du nuage		haut du nuage
Altitude (m)	0	800	1500	2000
Température (°C)	15	9,8	5,25	2
Densité de l'air	1,21	1,11	1,02	0,97
Vol rectiligne				
Vitesse (m/s)	20,00			
Mach	0,059	0,059	0,060	0,060
Cl rectiligne	0,101	0,111	0,120	0,127

Vol en virage				
Rayon de virage (m)	100			
Angle de roulis (°)	22,18			
Facteur de charge	1,08			
Cl virage	0,109	0,120	0,129	0,137

Vol longitudinal	Palier	Montée	Descente
Vitesse verticale (m/s)	0	3	-3
Angle de montée (°)	0,000	8,531	-8,531
ΔF (N)	0,000	209,217	-209,217
Direction ???			
Angle de dérapage			
Moment de roulis			

Hypothèses
le déplacement dans le plan longitudinal (montée/descente) ne se fait qu'à l'aide de la poussée
le maintien de la portance en virage se fait grâce à une modification du Cl (incidence)

paramètres fixés

FIGURE 2.7 – onglet ParamAVL du livrable Excel

2.3 Amélioration possible : inclure un modèle aérologique

Notre modèle de mission reste assez simpliste et plusieurs améliorations pourraient être mise en œuvre afin d'obtenir un profil de mission qui se rapproche davantage de la réalité. Par exemple, nous avons considéré que le modèle de vent était constant en direction et en norme quelle que soit l'altitude de notre drone. Ceci est très rarement le cas en réalité ; nous pourrions implémenter un modèle un peu plus proche de la réalité en imposant, par exemple, que le modèle de vent suive une loi linéaire en fonction de l'altitude.
Au niveau de notre fichier $Specifications\_mission.xsl$, nous pourrions ainsi définir le vent au niveau du sol et son orientation, ainsi que le gradient de vent en fonction de l'altitude. Puis, lorsque nous calculons les différentes vitesses et altitudes en fonction de la phase de mission dans l'onglet Cas de vol, nous pourrions prendre en compte l'altitude du drone par pas de 500m (par exemple) afin de recalculer à chaque fois la valeur du vecteur vent et son influence sur notre profil de mission. Ce modèle météorologique pourrait également être complexifié (variations non linéaires du vent, changement d'orientations, etc).
Enfin, nous pourrions mettre davantage en évidence dans notre fichier quand les conditions météorologiques ne sont pas favorables au bon déroulement d'une mission du drone. Pour l'instant nous savons que le range du drone est limité à 6km, mais il n'y a pas de limitations implémentée dans le fichier pour vérifier que le drone peut bien effectuer sa mission sans sortir de cette limite malgré le vent. Il serait donc judicieux de mettre en place cette limitation.

Chapitre 3

Amélioration du code

3.1 Présentation de la structure du code

Le programme d'optimisation est composé de blocs mono-disciplinaires développés en grande partie par M. Bouhoubeiny et d'un optimiseur chargé de connecter ces blocs entre eux et de d'optimiser un ou des paramètres. Le code est développé dans le langage *Python*.

Ces blocs mono-disciplinaires sont connectés entre eux dans un certain ordre par différents paramètres. Ils sont au nombre de 4 :

1. **Dimensionnement géométrique** : calcul des dimensions principales du drone.
2. **Dimensionnement de la structure** : calcul des masses de la structure du drone à partir des dimensions de l'appareil.
3. **Calculs aérodynamiques** : calcul des coefficients aérodynamiques nécessaires pour accomplir les différentes parties de la mission. Ces coefficients sont calculés à l'aide d'un logiciel annexe : *AVL*.
4. **Évaluation de l'endurance** : à partir des coefficients aérodynamiques, calcul de l'endurance du drone : paramètre à maximiser.

Tous ces blocs sont détaillés dans la section suivante 3.2. Nous retrouvons la structure principale du code sur le schéma suivant :

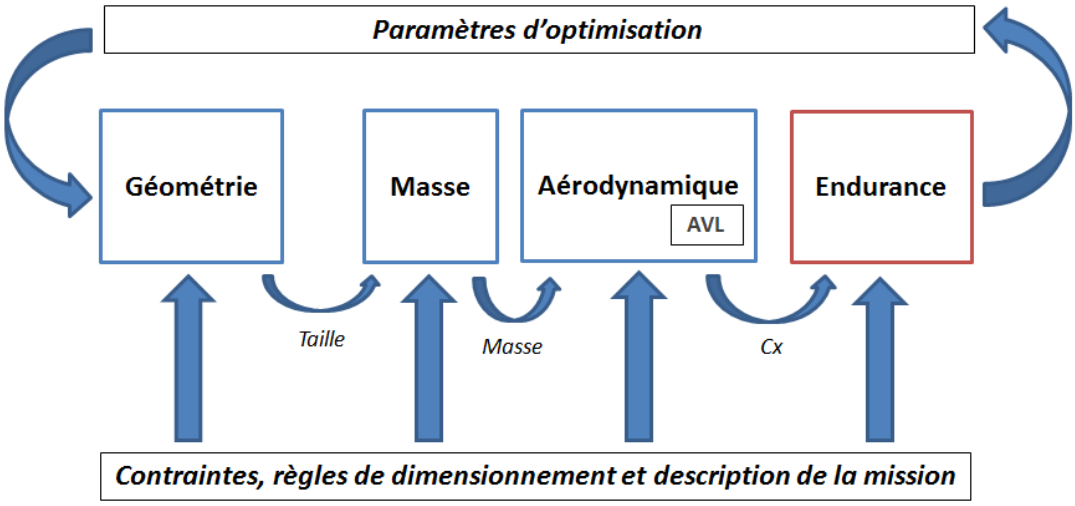


Figure 3.1 – Structure du code

Au total, le code est composé de 14 fonctions *Python* qui sont réparties de la façon suivante :

Catégorie	Nom du fichier Python	Description
Optimisation	skyscanner_group.py	Programme principale régulant toute l'optimisation
	skyscanner_group_without_opt.py	Idem mais sans la boucle d'optimisation
Paramétrage	common_inputs.py	Fichier de définition des principaux paramètres
Dimensionnement géométrique	aircraft_length.py	Calcul des principales dimensions du drone
Dimensionnement structurel	mass_model.py	Calcul des masses du drone
Coefficients aérodynamiques	generate_AVL_files.py	Génération des fichiers lisibles par AVL
	Aero_AVL.py	Procédure de calcul du coefficient aérodynamique de traînée
	avl_parameters.py	Lancement d'AVL pour les différents cas de vol
Endurance	pwr_concumption .py	Calcul de l'endurance du drone
Fonctions annexes	get_values.py	_
	getvalues.py	_
	gen_fuselage.py	_
	func_number_float.py	_
	func_spacing.py	_

FIGURE 3.2 – Liste des fichiers *Python*

3.2 Les blocs

3.2.1 Dimensionnement géométrique

Description géométrique du drone Le dimensionnement géométrique de notre drone est effectuée à partir d'un set de paramètres géométriques tels que le diamètre du fuselage ou l'épaisseur de l'aile (par exemple). Cette description paramétrique est complétée par des éléments non paramétriques :

- Les profils des différentes surfaces
- La géométrie précise des surfaces (elliptique/carrée, loi de vrillage...)
- La loi de section du fuselage

Dans le cas de notre programme, ces éléments sont choisis au préalable. On gardera cependant à l'esprit qu'il est toujours possible de décrire ces éléments de manière paramétrique (ex : paramétrisation NACA pour les profils) si un utilisateur voulait mener une optimisation plus poussée, sous réserve qu'il en ait les moyens (l'augmentation du nombre de paramètres complexifie le programme).

Rôle du programme de dimensionnement géométrique Le programme $aircraft\_length.py$ est chargé de la gestion des paramètres géométrique du drone précédemment exposés. Son rôle est de compléter la géométrie d'un drone dont on lui donne uniquement certains paramètres, et des contraintes à respecter. Nous allons maintenant détailler le fonctionnement de ce programme dans le cas du dimensionnement du Skyscanner.

Pour le Skyscanner, nous avons choisi de laisser libre la surface alaire de la voilure principale et son envergure. Notre programme $aircraft\_length.py$ choisit ensuite une longueur de fuselage et un empennage adapté à cette voilure principale. Les autres paramètres comme le dièdre ou la flèche sont choisis constants dans la version actuelle du programme par souci de simplicité.

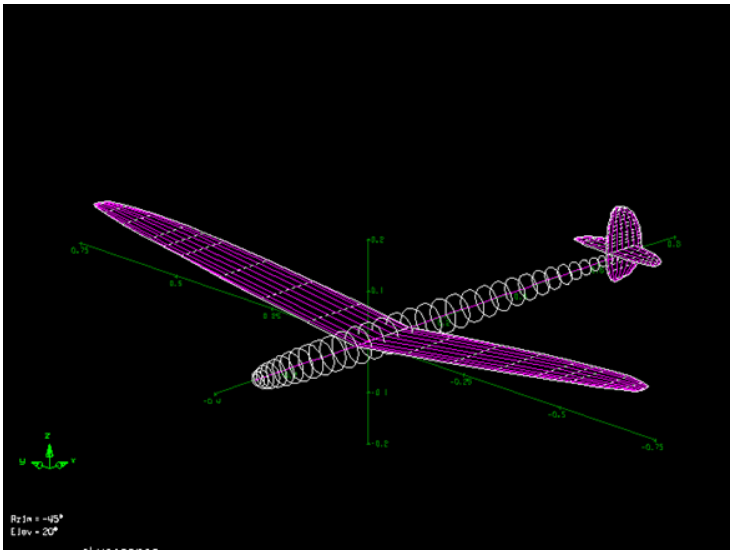


FIGURE 3.3 – Exemple générique d'un drone décrit par notre paramétrisation

Stabilité de l'aéronef L'ordre de grandeur des moments des efforts de la voilure principale sur l'aéronef est donné par :

$$\begin{aligned} M_{Gz} &= (F_{Xgauche} - F_{Xdroite})b = (qS_{aile})b \\ M_{Gy} &= F_Z d_{foyer-CG} = (qS_{aile})\sigma c_{aile} \end{aligned} \tag{3.1}$$

Ou $F_{Xgauche}$ et $F_{Xdroite}$ sont les efforts aérodynamiques exercés respectivement sur les ailes gauche et droite, $q = .5\rho V^2$, σ est la marge statique de l'aéronef et $c_{aile} = S_{aile}/b_{aile}$ la corde moyenne de l'aile.

Les moments stabilisateurs des empennages sont :

$$\begin{aligned} M_{VTP} &= qS_{VTP}2/3l_{fuselage} \\ M_{HTP} &= qS_{HTP}2/3l_{fuselage} \end{aligned} \tag{3.2}$$

Remarque : nous avons choisi de placer les ailes au premier tiers de la longueur totale du fuselage pour garder un choix fait dans les travaux de M. BRONZ dans [3].

Les nombres sans dimension définis dans l'équation 3.3 sont donc des indicateurs de la stabilité de l'avion.

$$\begin{aligned} \frac{M_{VTP}}{M_{Gz}} &= \frac{S_{VTP}l_{fuselage}}{S_{aile}b} \\ \frac{M_{HTP}}{M_{Gy}} &= \frac{S_{HTP}l_{fuselage}}{S_{aile}\sigma c_{aile}} \end{aligned} \tag{3.3}$$

Un moyen simple de fixer un critère de stabilité pour l'avion revient à fixer des valeurs pour ces deux ratios : c'est ce qui a été fait dans notre programme. On note que nous n'avons que 2 critères pour fixer 3 valeurs (S_{HTP}, S_{VTP} et $l_{fuselage}$, ce degré de liberté pourrait laisser place à une optimisation supplémentaire, mais dans un souci de simplicité nous avons choisi de garder une longueur de fuselage proportionnelle à l'envergure de l'avion, c'est la solution qui a été retenue dans [3], elle permet de se retrouver avec des drone à peu près homothétiques les uns part rapport aux autres.

Une fois cela fixé, on a finalement les lois de dimensionnement 3.4 :

$$\begin{aligned} l_{fuselage} &= 3/2C\_l_{HT}b_{aile} \\ S_{HTP} &= C_{HT}\frac{c_{aile}S_{aile}}{2/3l_{fuselage}} \\ S_{VTP} &= C_{VT}\frac{b_{aile}S_{aile}}{2/3l_{fuselage}} \end{aligned} \tag{3.4}$$

La géométrie précise de l'empennage est ensuite complètement déterminée par l'allongement de l'empennage choisi au préalable. Concernant la partie non paramétrique de la description des empennages :

— Un profil ht12 est utilisé pour les empennages
— Les empennages sont elliptiques, et sans vrillage

Pour aller plus loin Le programme actuel est avant tout conçu pour explorer différentes envergures et surface alaire (les deux paramètres majeurs pour décrire l'aéronef) de manière simplifiée. Il peut être tentant de vouloir rajouter de nombreux paramètres libres pour explorer plus finement les aéronefs imaginables (flèche, profils d'aile, etc...) car l'architecture actuelle du code permet de faire cela de manière assez souple. Il cependant garder à l'esprit que tout enrichissement du bloc géométrie se paie assez cher en complexité (ou fidélité) sur les autres blocs disciplinaires, notamment sur la partie structure (on se retrouve rapidement à dimensionner des objets très complexes). Vu l'état de maturité actuel de notre code et de sa plateforme, il nous semble qu'il reste suffisamment de choses à tester et améliorer dans d'autres parties du projet avant de se mettre à alourdir le code.

Nous recommandons donc de ne pas chercher à raffiner cette partie du code tant que le code et sa plateforme (OpenMDAO) ne sont pas arrivés à un niveau de maturité supérieur : cela alourdirait inutilement les travaux de développement.

3.2.2 Dimensionnement de la structure

Dans cette partie, supposons que le dimensionnement géométrique de l'avion est maintenant effectué. Nous allons chercher à évaluer les différentes masses de notre drone. Pour cela, nous allons imposer au drone de résister à des efforts maximaux, choisis en entrée du programme. Nous allons également limiter les déformations maximales des éléments de structure du drone.
Notre travail est basé sur le projet de dimensionnement du drone CORSICA effectué par Combier-Müller-Pomarat-Rivail dans [2]. Le fichier Python concerné dans notre programme est *mass_model.py*.

Hypothèses pour dimensionner la structure

Nous considérons dans notre programme la structure suivante du drone :

— fuselage en carbone constitué de deux tubes de diamètres d_1 et d_2 comme indiqué sur la figure ci-dessous. Le premier tube embarque la charge utile du drone et les batteries et soutient les ailes, tandis que le deuxième tube, de diamètre inférieur, soutient l'empennage arrière.

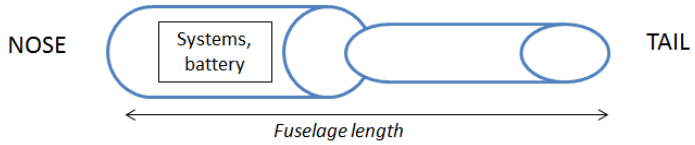


FIGURE 3.4 – Structure du fuselage

— voilure, plan horizontal et dérive constitués de fibres de carbones et d'un cœur en mousse, comme indiqué sur la figure ci-dessous.

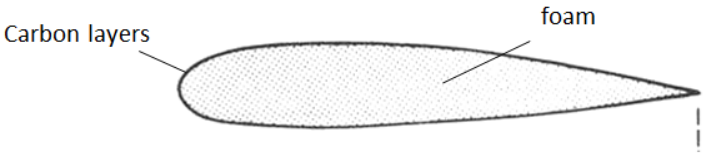


FIGURE 3.5 – Structure de l'aile

Efforts maximaux appliqués sur le drone

L'objectif est de concevoir un drone capable de résister à des efforts maximum qui sont définis comme des variables d'entrées de notre programme.
Les différents efforts maximum appliqués à notre drone sont les suivants :

— l'accélération maximum subie par le drone au décollage
Cette accélération maximale est une entrée numérique de notre fichier *common_inputs.py*.
— la portance maximale en croisière sur les ailes (calculée à partir d'un facteur de charge Nz maximal)
Cette portance maximale est calculée grâce à l'équation de sustentation $Portance = Poids$ multiplié par un facteur de charge Nz maximal. Ce facteur de charge est défini dans le fichier *common_inputs.py*.

$$Fz_{max} = m_{totale} g Nz_{max} \tag{3.5}$$

— la portance maximale en croisière sur l'empennage horizontal
La portance maximale sur l'empennage arrière est l'effort aérodynamique calculé à partir d'un Cz maximal et d'une vitesse maximale (VNE), définis dans *common_inputs.py*.

$$\begin{aligned} Fz_{htail} &= \frac{1}{2}\rho VNE^2 Cz_{max} S_{htail} \\ Fz_{vtail} &= \frac{1}{2}\rho VNE^2 Cz_{max} S_{vtail} \end{aligned} \tag{3.6}$$

— la traînée maximale appliquée sur les ailes
La traînée maximale appliquée sur les ailes est l'effort aérodynamique calculé à partir d'un Cx maximal et de la VNE, donnés dans le fichier *common_inputs.py*.

$$Fx_{max} = \frac{1}{2}\rho VNE^2 Cx_{max} S_{wing} \tag{3.7}$$

— la traînée maximale appliquée sur l'empennage arrière
La traînée maximale appliquée sur l'empennage arrière est l'effort aérodynamique calculé à partir d'un Cx maximal et de la VNE, donnés dans le fichier *common$_i$nputs.py*.

$$\begin{aligned} Fx_{htail} &= \frac{1}{2}\rho VNE^2 Cx_{max} S_{htail} \\ Fx_{vtail} &= \frac{1}{2}\rho VNE^2 Cx_{max} S_{vtail} \end{aligned} \tag{3.8}$$

— la différence maximale entre la poussée et la traînée qui provoque la traction du fuselage
Cette valeur est une entrée numérique de notre fichier *common_inputs.py*.

Nous allons dimensionner notre drone pour qu'il soit capable de résister à ces efforts. Également, nous imposerons que les déformations de certains éléments restent inférieures à une déformée maximale, afin de limiter la déformation du drone durant le vol, qui risqueraient d'entraver la bonne prise de mesures des capteurs.
Dans le cadre de notre drone Skyscanner, nous avons choisi les valeurs numériques suivantes :

Table 3.1 – Efforts maximaux utilisés pour le drone

Nz_{max}	4
Cx_{max}	0,1
Cz_{max}	2
VNE (m/s)	100
Accélération décollage (m/s^2)	60
Poussée moteur max (N)	20

La valeur de Cz_{max} choisie est une valeur usuelle (haute) pour les profils habituellement rencontrés en aéronautique. Pour le Cx_{max} nous avons pris une marge considérable, afin d'être sur de trouver des efforts au moins égaux à ceux rencontrés en réalité : le Cx_{max} étant assez sensible aux Reynolds rencontrés et au profil utilisé, nous avons pris une marge d'environ 100% sur les pires valeurs de Cx rencontrées (avant la crise de traînée). Si une étude plus précise est menée, il est possible de gagner plus d'un ordre de grandeur sur cet effort dimensionnant les surfaces portantes.

Les autres choix sont directement dépendant des conditions d'opération du drone : par exemple si le CNRM fait le choix d'exploiter le vol de gradient de manière agressive, les trajectoires obtenues vont mener

à des facteurs de charge et des vitesses air élevées (Nz_{max} et VNE), et donc nécessiter un renforcement de la structure du drone. Un projet étudiant sur ce sujet est en cours (fin en avril 2016), il permettra d'évaluer correctement les valeurs de Nz_{max} et la VNE. Les valeurs actuelles sont des ordre de grandeurs (majorés) issus du travail de V. Bonnin sur le vol de gradient [1].

L'accélération au décollage et l'effort maximal de compression sur le fuselage vont aussi dépendre des condition d'opération de l'aéronef : le CNRM compte catapulter ses drones, et nous avons pris une accélération d'environ 6g pendant le catapultage et une poussée moteur maximale de 20N (des valeurs utilisées par Combier et Al. dans [2]).

Calcul des différentes masses

Dimensionnement du fuselage Pour le fuselage on va considérer deux efforts dimensionnants. Tout d'abord, le moment maximum imposé par l'empennage arrière et la masse des systèmes embarqués et évalué, et le fuselage est dimensionné à cet effort pour respecter des contraintes de résistance et déformation maximale admissible :

On considère le moment maximum admissible par une couche de carbone qui vaut :

$$\begin{aligned} moment_{une\_couche} &= \frac{stress\_max\_layup}{\frac{1}{2}d_{fuselage}Igz_{une\_couche}} \\ Igz_{une\_couche} &= \frac{\pi}{64}(d_{fuselage}^4 - (d_{fuselage} - epaisseur_{une\_couche})^4) \end{aligned} \tag{3.9}$$

On calcule alors les moments maximum dus à la masse des systèmes embarqués et à la portance de l'empennage arrière ramenés à chaque tube du fuselage ; pour cela on calcule le moment théorique maximal :

$$\begin{aligned} moment_{max} &= moment_{masses} + moment_{portance\_empennage} \\ &= L_{tube}M_{embarquee}g + L_{fuselage}Fz_{empennage} \end{aligned} \tag{3.10}$$

On en déduit le nombre de couches théoriques pour que le fuselage résiste à ces efforts avec la formule :

$$nb_{couches} = E\left(\frac{moment_{max}}{moment_{une\_couche}}\right) + 1 \tag{3.11}$$

A partir du moment de flexion calculé précédemment, on va imposer une déformation maximale pour le deuxième tube du fuselage qui soutient l'empennage arrière. En effet, il possède un diamètre inférieur et est donc davantage soumis à des déformations du fait des efforts qui s'appliquent sur l'empennage (plan horizontal et dérive). On va donc renforcer la structure jusqu'à ce que la déformation soit limitée à 5% de la longueur totale du tube (critère de 5% empirique).

Le déplacement en bout de fuselage vaut :

$$f_1 = \frac{L_{tube2}^2\sqrt{moment_{vtp}^2 + moment_{htp}^2}}{3E_{carb}nb_{couches}Igz_{une\_couche}} \tag{3.12}$$

et le nouveau nombre de couches vaut alors :

$$nb_{couches} = E\left(\frac{L_{tube2}^2\sqrt{moment_{vtp}^2 + moment_{htp}^2}}{3E_{carb}f_1Igz_{une\_couche}}\right) + 1 \tag{3.13}$$

Enfin, on calcule dans chaque cas les masses finales de chacun des tubes du fuselage à l'aide des formules suivantes :

$$M_{tube} = nb_{couches}L_{tube}d_{tube}\pi\rho_{carbone} \tag{3.14}$$

Les fonctions Python associées sont $effHtpFuselagePoutre1$ et $effHtpFuselagePoutre2$.
Un second type de sollicitation mécanique est considéré, il s'agit des efforts de traction/compression sur le fuselage dus à la traînée et la poussée de l'avion. Pour cela, on calcule l'effort maximal admissible par une fuselage composé d'une seule épaisseur de couche de carbone, et on en déduit le nombre de couches nécessaires pour supporter cet effort.

On a ainsi, pour une couche de fibres :

$$Effort_{max\_fibre} = \pi d_{fuselage} epaisseur_{une\_fibre} contrainte_{max} \tag{3.15}$$

Le nombre de couches total est déduit par :

$$nb_{couches} = factTube(E\left(\frac{Poussee_{max}}{Effort_{max\_fibre}}\right) + 1) \tag{3.16}$$

avec $factTube$ un facteur de sécurité qui dépend du tube de fuselage étudié (empirique).
Les fonctions Python associées sont $effDragFusPoutre1$ et $effDragFusPoutre2$.

Dimensionnement de l'aile Pour le dimensionnement de l'aile, nous avons considéré 3 sollicitation dimensionnantes. La première que nous allons détailler est rencontrée lors du décollage de l'avion, par catapultage :

On souhaite que notre aile soit capable de résister à une accélération maximale. Les efforts d'accélérations sont répartis linéïquement le long de l'envergure de l'aile. Pour cela, on calcule le moment maximal supporté par la structure de l'aile lorsque celle-ci n'est composée que d'une couche de carbone et d'un cœur en mousse. Ensuite, on le compare avec le moment imposé par l'accélération maximale et on épaissit la structure tant qu'elle ne peut pas résister à cette accélération.

L'algorithme utilisé est le suivant :
$moment = 0$;
$resiste = 0$;
Tant que $resiste == 0\{$

$$\begin{aligned} moment_{une\_couche} &= \frac{1}{2} corde(3e_{wing} corde * e_{fiber} stress_{max}) \\ nb_{couches} &= E\left(\frac{moment}{moment_{une\_couche}}\right) + 1 \\ moment_{max} &= nb_{couches} moment_{une\_couche} \\ moment &= moment\_lineique\_acceleration \end{aligned} \tag{3.17}$$

si $moment < moment_{max}$ alors $resiste = 1\}$;
avec e_{fiber} l'épaisseur d'une couche de fibre et $stress_{max}$ la contrainte maximale admissible dans une fibre de carbone.

On calcule pour chaque cas la masse totale de l'aile via les formules suivantes :

$$\begin{aligned} M_{mousse} &= S_{wing} e_{wing} corde * factMousse \\ M_{aile} &= masse\_lin\_struc * b_{wing} + M_{mousse} \end{aligned} \tag{3.18}$$

Le facteur $factMousse$ est un facteur d'ajustement empirique, choisi ici à 2/3. La fonction Python associée est $momAccWing$.

Le second est la sollicitation que verra l'aile lorsque le facteur de charge de l'avion maximal. A cet instant, l'aile devra supporter la charge de $Nz_{max} M_{tot} g$ pour sustenter l'avion. Cela induit un moment de flexion auquel l'aile doit résister. Comme précédemment, on utilise à la fois un critère de résistance et de déformation maximale admissible.

Le moment de flexion à charge maximale est donné par :

$$moment_{max} = Position\_FzNz_{max} M_{totale} g \tag{3.19}$$

Puis, connaissant le moment maximal admissible par la structure quand elle ne possède qu'une couche de fibres de carbone, on épaissit la structure jusqu'à ce qu'elle résiste à cet effort.

$$\begin{aligned} moment_{une\_couche} &= \frac{1}{2} cordee_{wing} e_{fiber} stress_{max}) \\ nb_{couches} &= E\left(\frac{moment_{max}}{moment_{une\_couche}}\right) + 1 \end{aligned} \tag{3.20}$$

La fonction Python associée est $momLiftWing$.
Nous noterons que pour déterminer l'effort de portance, nous devons au préalable connaître la masse totale de l'avion, qui n'est pas encore définie puisque l'on est en train de calculer la masse des ailes. Ainsi, nous devons imposer une masse totale $M0$ initiale en argument de cette fonction, et nous bouclerons le programme masse jusqu'à obtenir une masse totale de notre drone constante à un gramme près.

Le dernier cas dimensionnement est celui rencontré par l'aile lorsque l'avion est à sa vitesse maximale autorisée. Dans ce cas, seul l'effort de traînée est pris en compte [1]. De même que précédemment on calcule le moment des efforts de traînée sur l'aile, et on dimensionne l'aile en conséquence avec les équations 3.21. Ici seule la résistance de l'aile a été considéré (pas de critère de déformation maximale admissible).

$$\begin{aligned} moment_{trainee} &= \frac{FxPos_{Fx}}{\frac{1}{2}corde} \\ effort_{max\_une\_couche} &= 3e_{wing}e_{fiber}stress_{max}corde \\ nb_{couches} &= E\left(\frac{moment_{trainee}}{effort_{max\_une\_couche}}\right)+1 \end{aligned} \tag{3.21}$$

La fonction Python associée est $effDragWing$.

Empennage arrière et dérive En ce qui concerne l'empennage arrière et la dérive, les masses sont déterminées en considérant les efforts aérodynamiques maximum qui s'appliquent sur ces surfaces (portance et traînée).

En ce qui concerne la portance maximale qui s'exerce sur ces éléments, on détermine le moment maximal exercé sur ces surfaces à l'aide du Cz_{max}. Pour la traînée, on utilise le Cx_{max}. Ces deux coefficients sont des entrées de notre programme, et leurs valeurs numériques sont stockées dans le fichier *common_inputs.py*. De plus, nous avons également limité la déformation de ces éléments à 20% de la longueur des cordes respectives de ces éléments (critère empirique). La manière de limiter ces déformations est calculée de la même façon que pour le fuselage.

Les fonctions Python associées sont $momLift\_Htp$ et $momLift\_Vtp$ pour les moments dus à la portance, et $effDrag\_Htp$ et $effDrag\_Vtp$ pour les efforts dus à la traînée.

Masse de la batterie Nous déterminons la masse de la batterie à partir de la valeur de la capacité de la batterie, selon la formule suivante :

$$M_{bat} = \frac{C_{bat}V_{bat}}{k_{batt}} \tag{3.22}$$

Où C_{bat} et V_{bat} sont respectivement la capacité et la tension nominale de la batterie, k_{batt} un coefficient défini par M. Bronz dans [3].

Masse des systèmes embarqués La masse des systèmes embarqués est fixe et est obtenue par la formule 3.23 :

$$M_{syst} = M_{payload} + M_{autopilot} + M_{sensors} + M_{modem} + n_{servo}M_{servo} + M_{cable} + M_{receiver} + M_{GSM} + M_{ESC} \tag{3.23}$$

Masse des moteurs La masse des moteurs est fixe et est obtenue par une formule empirique, à partir de la puissance de ce dernier :

$$M_{mot} = P_{mot}k_{mot} \tag{3.24}$$

Où k_{mot} est un coefficient évalué dans [3].

1. Il est surréaliste de dimensionner l'aile pour résister à la portance max à la VNE (cela correspondrait à un facteur de charge très élevé) : c'est au rôle du pilotage de limiter le facteur de charge maximal de l'avion.

Masse totale du drone Pour calculer la masse de chaque élément, nous calculons autant de masses différentes qu'il y a d'efforts qui s'appliquent sur cet élément à partir des formules précédentes. Puis, nous choisissons la masse qui nous permet de résister à l'ensemble de ces efforts, et donc le maximum des différentes masses.
La masse totale du drone est obtenue en sommant toutes les contributions des différentes masses, à savoir :

$$M_{total} = M_{wing} + M_{fuselage} + M_{derive} + M_{empennage} + M_{batterie} + M_{systemes} + M_{moteurs} \tag{3.25}$$

Résultats obtenus

Nous avons calculé les masses obtenues dans le cas d'un drone de surface alaire de $0.24m^2$, d'envergure de 2m, et de diamètre et de longueur de fuselage équivalente (respectivement 0.07m et 1.364m). La masse des systèmes embarqués vaut dans ce cas 0.733g.
Le tableau ci-dessous compare les masses obtenues avec les formules empiriques de M. Bronz dans [3] et notre programme.

Element structurel	Masse de notre programme (kg)	Masse donné par MURAT (kg)	Ecart en %
Fuselage	0,09	0,044	*105%*
Aile	0,174	0,17	*2%*
Plan horizontal	0,027	0,004	*575%*
Dérive	0,001	0,005	*-80%*
Systèmes	0,733	0,733	*0%*
Batterie	0,654	0,654	*0%*
Moteurs	0,07	0,07	*0%*
Drone entier	1,749	1,68	*4%*

Nous constatons que les ailes sont correctement dimensionnées par rapport aux valeurs données par M. Bronz dans [3].
Le fuselage est plus lourd, ce qui s'explique par les valeurs des efforts maximaux qui s'appliquent sur le fuselage qui sont largement surévalués (notamment au niveau des bras de levier, qui sont maximaux). De plus, il serait intéressant de réduire les facteurs de sécurité k_{tube} qui sont peut être un peu surestimés.
Au niveau de l'empennage arrière, nos valeurs diffèrent de celles de [3] car nous avons une approche complètement différente de celle de M. Bronz. Notre approche est basée sur des calculs d'efforts aérodynamiques maximum et la structure même de l'empennage arrière est modifiée (il s'agit maintenant d'un empennage à structure de fibres de carbone et d'un cœur en mousse). Aussi il est difficile de comparer nos valeurs à celles trouvées dans [3] qui sont plutôt basé sur des statistiques et ne prennent pas en compte les effort réels rencontrés par l'avion.

Recommandations

Afin d'améliorer cette partie nous proposons plusieurs axes d'amélioration :

Concernant l'amélioration du modèle de masse Un travail peut s'effectuer pour rendre ce modèle de masse encore plus proche de la réalité en travaillant sur :
- la prise en compte des efforts au niveau du fuselage et les facteurs de sécurité
- le calcul exact de la masse de la mousse
- la prise en compte de la poussée et de la portance linéïquement et non pas comme un effort moyen sur les ailes et l'empennage

Concernant le centrage dans AVL Pour la partie aérodynamique, AVL n'utilise actuellement pas nos valeurs de masses pour calculer le centrage de l'appareil.
Afin de prendre en compte notre modèle, il faudrait créer un fichier *.mass* qui comporterait les différents éléments structuraux du fichiers *mass_model.py* (tubes, ailes, batteries, etc) ainsi que les masses, positions et inerties de chacun de ces éléments, calculés par notre programme. Des exemples de ce genre de fichiers sont en opensource et donc facile d'accès.
Enfin, afin de calculer le centrage final de notre avion, il serait intéressant d'autoriser le déplacement de

l'un de nos éléments le long de l'axe de l'appareil, par exemple la masse de la batterie ; puis, de reboucler sur cette position de batterie jusqu'à obtenir la stabilité de notre drone.
Enfin, pour permettre ce déplacement, il est nécessaire de modifier le fichier *aero.avl* afin d'effectuer ce déplacement dans le fichier en *.mass* et non dans le fichier en *.avl*.

3.2.3 Calculs aérodynamiques

Les dimensions principales et la masse du drone sont maintenant connues. Elles ont été calculées dans les blocs précédents (voir section 3.2.1 et section 3.2.2). Afin d'estimer l'endurance du drone, il est nécessaire de calculer sa traînée. Pour cela, nous utilisons un logiciel de calcul d'écoulement potentiel appelé *AVL*.

AVL *AVL* est un programme d'analyse aérodynamique et de dynamique du vol pour des configurations arbitraires d'aéronefs rigides. Le logiciel emploie un modèle étendu de vortex pour les surfaces de portance et un modèle de corps élancés pour les fuselages et nacelles. Des cas de vol non-linéaires peuvent être spécifiés. L'analyse de dynamique du vol permet une linéarisation complète du modèle aérodynamique dans n'importe quel état de vol.

L'utilitaire *AVL* est lancé automatiquement par le programme. Pour cela, il est nécessaire de :

1. Écrire dans des fichiers lisibles par *AVL* les caractéristiques géométriques du drone et les caractéristiques des cas de vol.
2. Lancer *AVL* et faire lire au logiciel les fichiers écrits à l'étape précédente.
3. Calculer les coefficients aérodynamiques du drone pour chaque cas de vol.
4. Faire écrire à *AVL* les résultats dans un fichier.
5. Lire le fichier des résultats pour en déduire le coefficient de traînée C_x.

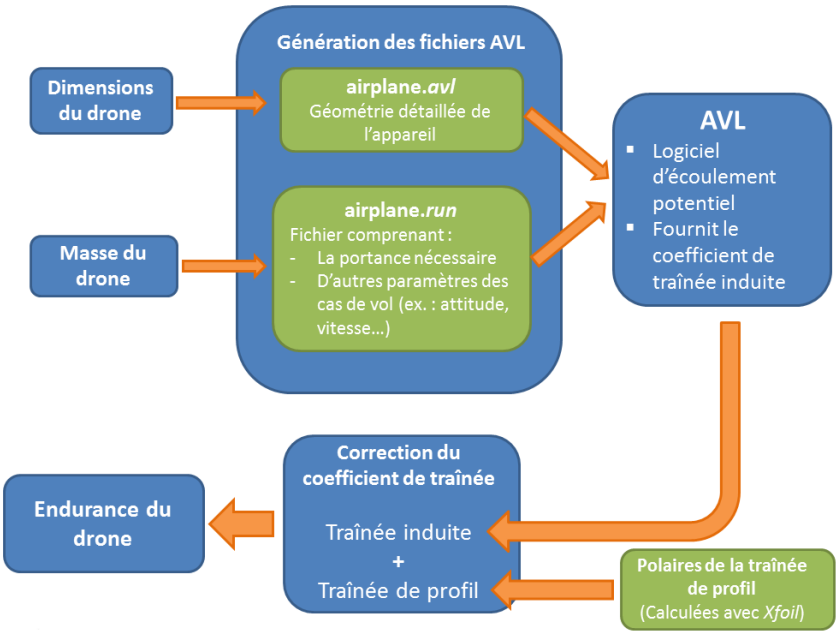


FIGURE 3.6 – Calcul du coefficient de traînée du drone

Ecrire dans des fichiers lisibles par *AVL*

Cette fonction est réalisée par le fichier *generate_AVL_files.py*. Ce fichier prend en paramètre les dimensions du drone, sa masses ainsi que des paramètres définis dans le fichier *common_inputs* comme les surfaces de référence.

Pour fonctionner correctement, *AVL* a besoin de 2 fichiers :
- Un fichier décrivant précisément la géométrie du drone : *airplane.avl*.
- Un fichier décrivant les cas de vol auxquels le drone sera confronté : *airplane.run*.

airplane.avl Jusqu'à présent, la description géométrique du drone se limitait à ses dimensions générales. Mais *AVL* nécessite une description bien plus détaillée de toutes les surfaces portantes. Le fichier *airplane.avl* détaille donc toutes les composantes du drones : fuselage, ailes et empennage. Toutes ces composantes sont décomposées en sections réparties à intervalles réguliers. La normalisation de ce fichier *.avl* est détaillée dans la documentation liée au logiciel. Cette documentation est disponible à : `http://web.mit.edu/drela/Public/web/avl/avl_doc.txt`.

airplane.run Ce fichier décrit les caractéristiques de chaque cas de vol comme l'inclinaison de l'appareil, son taux de montée, sa vitesse, sa masse... Comme pour le fichier précédent, son contenu est décrit dans la documentation de *AVL* disponible sur `http://web.mit.edu/drela/Public/web/avl/avl_doc.txt`.

Lecture des fichiers par *AVL*, calcul des coefficients et renvoi des résultats

La lecture des fichiers *airplane.avl* et *airplane.run*, le lancement des calculs aérodynamiques, l'écriture des résultats dans un fichier et la lecture de ce dernier par notre programme est réalisé par le même fichier : *Aero_AVL.py*.

Des fonctions secondaires situées dans le fichier *avl_parameters* sont chargées de lancer *AVL* pour chaque cas de vol. Nous avons 6 cas de vol : *AVL* est donc lancé 6 fois. En réalité, *AVL* est lancé 7 fois car le logiciel est lancé une première fois afin de déterminer la nouvelle position du centre de gravité de l'appareil et celle du calage de l'aile. Ces positions sont calculées afin de stabiliser l'avion durant la majeur partie du vol : donc pour le cas de vol majoritaire.

AVL écrit alors les résultats dans un fichier appelé *avl_forces_cf.dat*. Ce fichier est lu toujours dans le fichier *Aero_AVL.py* à l'aide d'une fonction secondaire située dans le fichier *get_values.py*. Grâce à cette fonction, il est possible de récupérer la valeur du coefficient de traînée C_x.

Correction du coefficient de traînée C_x

Attention, le coefficient de traînée C_x calculé par *AVL* n'est pas correct. En effet, *AVL* est un logiciel d'écoulement potentiel, il ne calcule donc que le coefficient de traînée induite. Il manque la composante de la traînée de profil. Si ce problème n'est pas corrigé, alors la traînée du drone dépend peu de l'incidence de ce dernier : le logiciel d'optimisation va chercher à diminuer la taille (et donc la masse) des ailes au maximum et faire voler le drone à des incidences irréalistes.

Pour ajouter le coefficient lié à la traînée de profil, nous avons utilisé des polaires provenant d'un autre logiciel : *Xfoil*. Ce logiciel fournit les évolutions du coefficient de traînée de profil en fonctions de l'incidence de l'aile. Pour les incidences qui nous intéresse, ces évolutions sont linéaires. Par contre, les coefficients de ces fonctions linéaires évoluent quadratiquement par rapport au nombre de Reynolds. En faisant une interpolation de ces évolutions au second ordre, nous parvenons à définit une loi qui fournit le coefficient de traînée de profil $C_{x_{profil}}$.

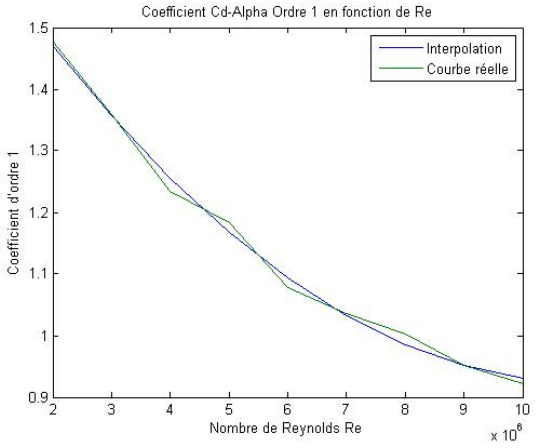


FIGURE 3.7 – Interpolation pour le coefficient d'ordre 1 a_1

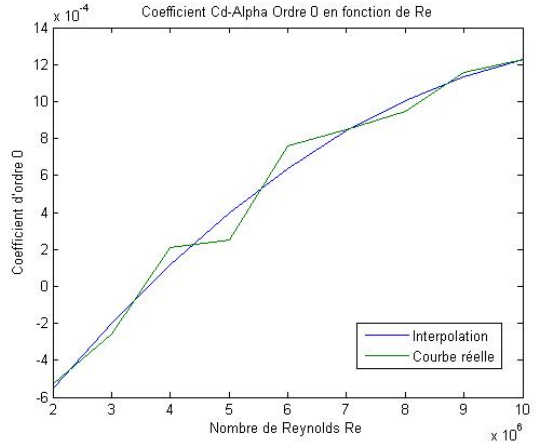


FIGURE 3.8 – Interpolation pour le coefficient d'ordre 0 a_0

$$C_{x_{total}} = C_{x_{induite}} + C_{x_{profil}} \tag{3.26}$$

avec

$$C_{x_{profil}} = |a_1 * (\alpha_{aile} + \alpha_{calage}) + a_0| \tag{3.27}$$

3.2.4 Évaluation de l'endurance

Une fois les caractéristiques aérodynamiques et la masse de l'avion connue, on peut évaluer l'endurance de l'avion sur le profil de mission considéré. Cette fonction est réalisée par le fichier *pwr_ consumption.py*.

La méthode pour évaluer la puissance repose sur un bilan de puissance : on évalue la consommation électrique pour chacun des cas de vol considéré et, connaissant la répartition des cas de vol dans la mission de l'avion, on déduit l'endurance de l'avion.

L'équation 3.28 fait le bilan de puissance sur l'avion pour le cas de vol i :

$$Pm^i = D^i U^i_{air} + M_{drone} g U^i_z \tag{3.28}$$

Où Pm^i est la puissance motrice nécessaire au maintient du vol, $D^i = C^i_d U^i_{air}$ la traînée de l'avion, et U^i_z la vitesse verticale du drone pour le cas de vol i.

En connaissant la consommation électrique des systèmes de l'avion, et le rendement des différents systèmes, on détermine la puissance prélevée sur la batterie pendant le vol dans le cas i.

$$P^i_{elec} = \frac{Pm^i}{\eta_{helice}\eta_{moteur}\eta_{esc}\eta_{batterie}} + \frac{P_{avionique} + P_{capteurs}}{\eta_{batterie}} \tag{3.29}$$

Dans la version actuelle du code, seule Pm^i est réévaluée à chaque passage dans la boucle d'optimisation, les autres paramètres décrivant le système électrique du drone sont estimés au préalable et choisis constants dans la le programme d'optimisation.

Maintenant que la consommation électrique de l'avion est connue pour chacun des cas de vol, on peut calculer l'endurance du drone à partir de son profil de mission. La version actuelle du code se base sur une répartition donnée des cas de vol pour calculer une consommation électrique moyenne :

$$< P_{elec} >= \sum_{i=1}^{N} a^i P^i_{elec} \tag{3.30}$$

Où a^i est la part du temps de mission passé dans le cas de vol i.

L'endurance T du drone est alors déterminée par la capacité de batterie :

$$T = \frac{E_{bat}}{< P_{elec} >} \tag{3.31}$$

Où $E_{bat} = C_{bat}V_{bat}$ désigne la capacité énergétique de la batterie.

Ce programme donne l'endurance d'un drone sur un profil de mission qui ne dépend pas de sa durée totale. Sa structure est suffisamment modulaire pour s'adapter facilement à des adaptations ou raffinements divers. Par exemple, pour le Skyscanner, il peut être intéressant de maximiser non pas l'endurance totale, mais le temps de vol utile (on retire le temps mis pour se rendre et revenir du nuage). Pour cela, on commence par évaluer l'énergie nécessaire pour se rendre et revenir du nuage :

$$E_{trajet} = P_{aller}t_{aller} + P_{retour}t_{retour} \tag{3.32}$$

Et on reprend le calcul précédent en remplaçant E_{bat} par $E_{bat} - E_{trajet}$ dans (3.31). Le temps T ainsi calculé correspondra alors bien au temps passé dans le nuage (celui qui intéresse le CNRM). Cette adaptation faite pour le cas du Skyscanner n'est qu'un exemple : cette partie endurance est facilement adaptable et peut évoluer en fonction de ce à quoi s'intéresse le CNRM et ce qu'il compte faire faire au drone.

3.3 Optimisation

3.3.1 Formulation du problème

Avant de pouvoir utiliser un optimiseur, il faut formuler correctement le problème d'optimisation. Un problème d'optimisation sous contrainte se définit de la manière suivante :

$$\begin{array}{l} \max_X f(X) \\ g_i(X) \leq 0 \end{array} \tag{3.33}$$

Où la fonction f est la fonction objectif et les fonctions g_i sont les fonctions décrivant les contraintes du problème. X est un vecteur de dimension N appartenant à l'espace d'optimisation. Dans le cas du Skyscanner, la fonction f est l'endurance du drone, et le vecteur X représente l'ensemble des variables sur lesquelles nous réalisons l'optimisation (ex : envergure, surface alaire...). Les fonctions g_i représentent des limitations que le drone doit satisfaire (limitation en masse, en charge alaire...).

Le premier travail consiste à s'assurer que la fonction f est bien définie, à savoir, que une fois que le vecteur X est donnée, il définit bien un drone unique et une endurance. C'est ce qui a été fait dans les blocs mono-disciplinaires :

- Les blocs géométrie et structure définissent un drone unique à partir d'un nombre réduit de paramètres (ex : envergure et surface alaire) et d'un ensemble de lois de dimensionnement
- Les blocs aérodynamique et endurance évaluent la fonction objectif pour ce drone

Notre fonction objectif f est donc bien définie par l'enchaînement de programmes mono-disciplinaires. Il en va de même pour les fonctions contraintes g_i qui sont elles aussi des fonctions définies à partir de paramètres évalué dans les blocs mono-disciplinaires.

Remarque : Si l'espace d'optimisation est modifié (ajout ou retrait d'une variable d'optimisation), ou la description du drone enrichie (ex : prise en compte de la répartition des masses), il faut reprendre les blocs géométrie et structure et s'assurer qu'ils définissent bien un drone de manière unique : tous les paramètres décrivant la géométrie du drone ou sa mission doivent être définis correctement. Dans notre programme, il y a 2 types de définition de ces variables :

— Si le paramètre fait partie des variables d'optimisation, il est définit par l'algorithme d'optimisation, dans le programme principal *skyscanner_group.py*

— Si le paramètre ne fait pas partie des variables d'optimisation il est calculé dans les programmes mono-disciplinaires, il est une fonction des paramètres d'optimisation et de constantes prédéfinies dans le programme *common_inputs.py*

3.3.2 Algorithmes d'optimisations

Maintenant que nous avons vérifié que le problème d'optimisation était bien posé, nous pouvons utiliser les algorithmes implémentés dans OpenMDAO pour le résoudre.

Pendant la quasi-totalité du projet nous avons utilisé l'algorithme d'optimisation COBYLA. Ce choix a été motivé par la simplicité de l'algorithme tant sur son principe dans sa mise en œuvre. COBYLA est un algorithme à direction de descente ne nécessitant pas de calcul de gradient : à chaque itération, une direction est choisie et le set de variables libres progresse d'une certaine distance dans cette direction, la distance parcourue à chaque itération diminue au cours de l'avancée de l'algorithme et ce dernier s'arrête lorsque cette distance est passée sous un seuil choisit à l'avance. On comprend rapidement que cet algorithme peut facilement se laisser attraper par des minima locaux et que plusieurs optimisations avec des conditions initiales différentes doivent être lancées pour obtenir un résultat pertinent. Nous recommandons cependant l'usage de cet algorithme car il a l'avantage d'être facilement traçable pendant son travail : cela facilite grandement la détection de problèmes dans le code (on suit la progression de l'algorithme pas à pas, ce qu'on ne pourrait pas faire aussi facilement avec un algorithme génétique).

Une autre raison de délaisser pour l'instant l'usage d'algorithme d'optimisation plus lourds est que leur implémentation se fait indépendamment du reste du programme : il sera facilement possible de le faire plus tard, lorsque le reste du code sera plus mature. Enfin, les algorithmes tels qu'ils ont été implémentés dans la version 1.4 de OpenMDAO ne sont pas terminés et sont largement sous-performants par rapport aux algorithmes théoriques. La plateforme OpenMDAO n'étant pas encore assez mature pour proposer des algorithmes sans gradient efficaces, nous conseillons de continuer à faire des travaux sur le reste du code (architecture des différents blocs), pour laisser le temps au code et à sa plateforme OpenMDAO de gagner en maturité.

3.4 Validation

Une fois les blocs monodisciplinaires écrits et le problème d'optimisation posé, on peut effectuer la validation du programme. Celle-ci s'organise en deux parties :

— l'obtention des drones correspondant au cahier des charges du CNRM,

— la vérification de la pertinence du programme, par des lois physiques et des comparaisons avec la littérature

3.4.1 Drones obtenus pour la mission du CNRM

Le cahier des charges du CNRM (voir section 2) détaillait trois profils de mission, et imposait des spécifications d'opération et de performance pour les drones. Nous avons traduit ces exigences en contraintes dans le problème d'optimisation :

— masse maximale du drone : 2.5 kg

— envergure maximale du drone : 1.5 m

De plus, plusieurs paramètres sont fixés en entrée :

— charge utile : 0.5 kg

— VNE : 100 m/s

— taux de montée : 1.5 m/s

Le choix a été fait de prendre comme variables d'optimisation :

— deux paramètres géométriques : la surface alaire et l'envergure

— un paramètre d'opération : la vitesse de croisière
— un paramètre d'endurance : la capacité de la batterie

L'optimiseur pourra faire varier ces paramètres, en veillant toutefois à ce qu'ils restent positifs et satisfaissent les contraintes ci-dessus.

La fonction objectif est la maximisation de l'endurance de notre drone.

Après optimisation, trois drones sont obtenus, qui optimisent l'endurance pour chacune des trois missions :

Caractéristiques	Mission 1	Mission 2	Mission 3
Masse totale (kg)	2.19	2.48	2.5
Surface alaire (mètre au carré)	0.1	0.16	0.11
Envergure (m)	1.5	1.43	1.45
Masse de la batterie (kg)	1.17	1.52	1.51
TAS (m/s)	32.9	34.6	35.2
Endurance (h)	7.7	3.97	4.11

Ces trois drones répondent au cahier des charges imposé par le CNRM.

Mission 1

Le drone obtenu est représenté sur la figure suivante :

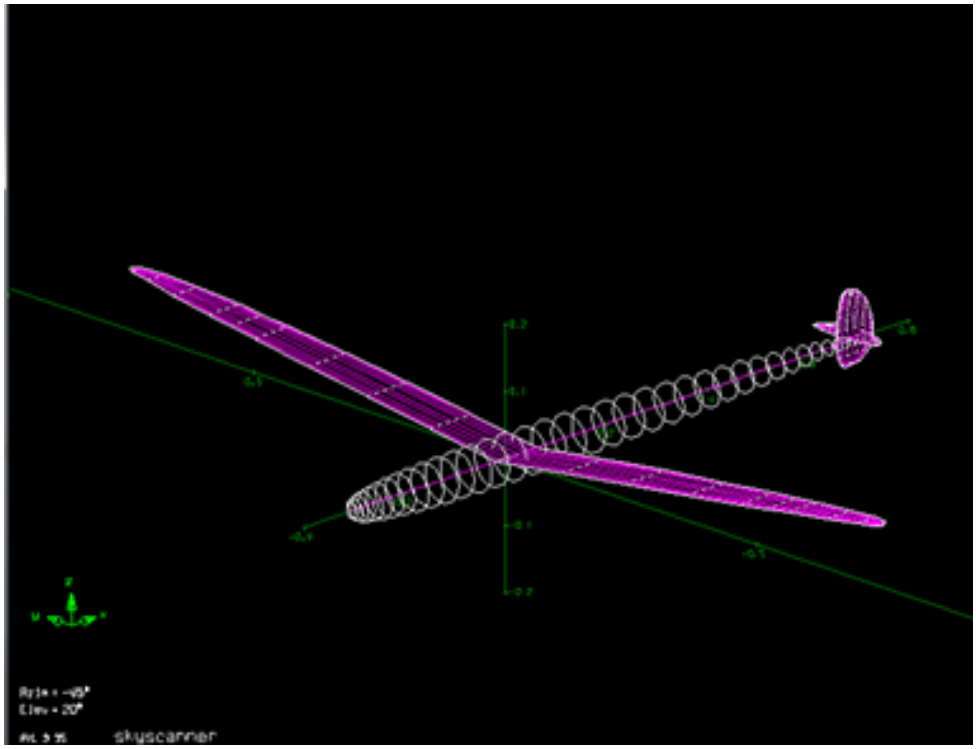


FIGURE 3.9 – Drone designé pour la mission 1

Pour pouvoir porter une charge utile de 0.5 kg, on utilise l'envergure maximale permise par le cahier des charges. Nous remarquons que la masse totale est autour de 2.2kg, donc inférieure de 300 grammes à la contrainte (fixée à 2.5 kg). On peut alors se demander pourquoi toute cette masse n'a pas été convertie en masse supplémentaire de batterie pour augmenter l'endurance. Mais si tel avait été le cas, la surface alaire aurait dû être plus grande, et donc la traînée sensiblement augmentée. L'optimum d'endurance est donc trouvé pour cette masse de drone.

Mission 2 et 3

Les missions 2 et 3 sont très différentes de la première, en ce sens que la part des montées et descentes à l'intérieur du nuage est beaucoup plus importante. Étant donné le profil très proche des deux missions, on obtient deux drones aux caractéristiques très proches :

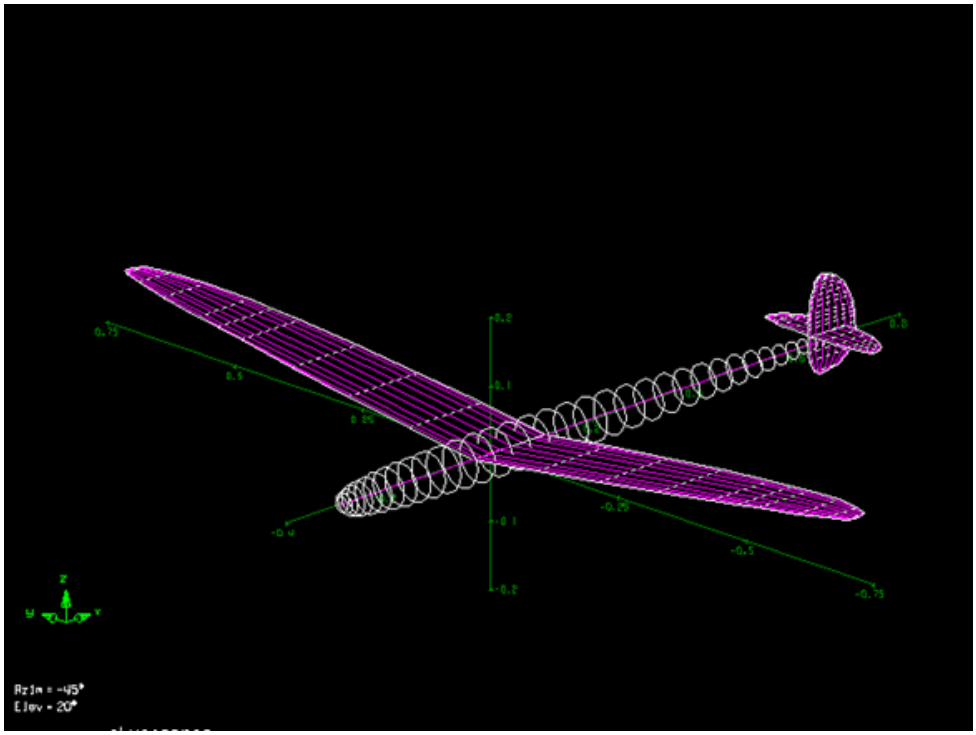


FIGURE 3.10 – Drone designé pour les missions 2 et 3

La masse obtenue est cette fois égale à la masse maximale autorisée. En effet, pour pouvoir faire toutes les montées, le drone va devoir embarquer le plus d'énergie possible, et c'est pourquoi la masse de la batterie occupe toute la masse restée disponible. De plus, les caractéristiques géométriques (envergure et surface alaire) restent assez limitées, car le drone doit atteindre un certain taux de descente qu'il n'arrivera pas à réaliser si la surface alaire est trop grande.

En résumé, lors de la première mission, le drone vole grâce à sa géométrie alaire car le vol en palier occupe une place prépondérante de la mission. Et dans les deux autres missions, le drone vole et réalise les cycles de montée et de descente grâce à l'énergie qu'il embarque.

3.4.2 Les lois physiques

L'objet de la seconde partie de la validation est la vérification de la pertinence de notre programme d'optimisation. Dans un premier temps nous vérifierons que notre programme donne des résultats physiquement logiques, et dans un deuxième temps nous essaierons de retrouver des valeurs proches des drones du commerce.

Nous avons pris comme référence le drone obtenu sur la mission 1. A partir de cette référence, nous modifions un à un certains paramètres d'entrée, puis nous observerons si les nouveaux résultats obtenus correspondent aux tendances physiques.

Impact de la vitesse de croisière

Nous fixons ici une vitesse TAS inférieure à celle de la mission 1, et nous optimisons sur la géométrie seule :

TAS (m/s)	30	20
Envergure (m)	1,41	1,5
Surface alaire (m^2)	0,14	0.33
Capacité batterie	11	11
Endurance (h)	5,36	5,31

FIGURE 3.11 – Impact de la vitesse de croisière

On obtient bien une augmentation de la surface alaire, ce qui est logique au vu de l'équation de sustentation :

$$mg = \frac{1}{2}\rho V^2 C_z S \tag{3.34}$$

Une diminution de V, tous paramètres égaux par ailleurs, engendre une augmentation de S.

Impact de la batterie

On augmente la capacité de la batterie (paramètre d'entrée fixé) et on observe le résultat sur les dimensions géométriques et l'endurance. On libère aussi la contrainte de masse maximale.

Capacité de la batterie	11	15
Envergure (m)	1,41	1,5
Surface alaire (m^2)	0,14	0,17
TAS	30	30
Endurance	5,36	5,12

FIGURE 3.12 – Impact de la batterie

Une augmentation de la capacité de la batterie provoque une augmentation des dimensions géométriques (surface alaire et envergure), ce qui est logique puisque la batterie est plus lourde, et il y a donc plus de masse à porter (Equation de sustentation). De plus, l'augmentation de la géométrie du drone va engendrer une hausse de la traînée, et donc une baisse de l'endurance. On voit bien là qu'une augmentation de la batterie de signifie donc pas toujours une amélioration de l'endurance.

Impact de la charge utile

On modifie seulement ici la charge utile, qui est une donnée fixée du problème d'optimisation.

Charge utile (kg)	0,5	2,5
Envergure (m)	1,5	1,5
Surface alaire (m^2)	0,168	0,24
TAS (m/s)	31,45	31,1
Endurance (h)	6,71	2,47
Masse totale (kg)	2,49	3,96

FIGURE 3.13 – Impact de la charge utile

Plus la charge utile est lourde, plus le drone aura des dimensions importantes. On se retrouve donc dans le cas précédent, ce qui conduit à une baisse de l'endurance (même à capacité de batterie laissée libre et sans contrainte de masse maximale).

Impact de la VNE
La VNE, où vitesse à ne jamais dépasser, est également un paramètre fixé au début de l'optimisation. Quel est son impact sur les résultats de la boucle d'optimisation ?

VNE (m/s)	100	50
Envergure (m)	1,5	1,5
Surface alaire (m^2)	0,168	0,158
TAS (m/s)	31,45	30,05
Endurance (h)	6,71	5,94
Masse totale (kg)	2,49	1,99

FIGURE 3.14 – Impact de la VNE

Lorsque la VNE diminue, cela veut dire que le drone est designé pour une vitesse maximale plus faible. Il sera donc soumis à des efforts aérodynamiques moins importants. En conséquence les contraintes structurelles seront plus faibles, et la masse de la structure diminuera. On aura donc une surface alaire plus faible.

L'étude de la validation des lois de la mécanique du vol peut ainsi être résumée dans le tableau suivant :

Variations	Effects
↘ TAS	↗ S_wing
↗ Battery characteristics	↗ Geometry dimensions ↘ Endurance
↗ Payload	↗ S_wing
↘ VNE	↘ Mass ↘ S_wing

FIGURE 3.15 – Résumé de la validation par la physique du vol

Notre programme fournit donc un résultat correspondant aux lois de la physique.

3.4.3 Comparaison aux drones du commerce

Enfin, afin d'étendre la validation, nous tentons de comparer les résultats de notre optimiseur avec un drone du commerce : le DT18 de Delair-Tech. Il s'agit d'un drone à voilure fixe, utilisé pour des missions d'observation civiles. Nous fixons les paramètres du DT18 connus :

— la vitesse de croisière : 14 m/s
— la charge utile : 0.5 kg

Les autres variables d'optimisation sont laissées libres et nous ne fixons aucune contrainte de masse ou d'envergure. Ainsi nous obtenons les résultats suivants :

Characteristics	DT18	Our drone	Difference
Total mass(kg)	2	1,67	16%
Wing surface(m^2)	0,35	0,62	77%
Span(m)	1,8	1,85	2,7%
TAS (m/s)	14	14	/
Endurance (h)	2	1	50%

FIGURE 3.16 – Comparaison avec le drone DT18 de Delair-Tech

On retrouve des résultats proches pour la masse et l'envergure. En revanche, les différences sont plus marquées pour la surface alaire et l'endurance. Cela est sans doute dû au type de mission différent (profil linéaire pour le DT18, beaucoup plus de virages pour notre drone Skyscanner), qui conduit notre drone à avoir une surface alaire plus grande. De plus, on ne connaissait pas non plus le type et la masse de la batterie embarquée à bord du DT18.

Chapitre 4

Gestion de projet

Ce projet ayant une forte vocation à enseigner aux étudiants les principes de la gestion de projet, le chapitre suivant traitera de ce sujet. Nous passerons en revue différentes notions et expliciterons comment elles ont été traitées au sein de notre projet afin de faciliter son organisation.

4.1 Rappel du cadre du projet

Comme détaillé dans l'introduction de ce rapport, le projet Skyscanner est un projet de recherche scientifique visant à développer une flotte de drone. Ces drones auront pour mission de suivre l'évolution des nuages et effectuer des mesures météorologiques autour de ces derniers. Le projet est constitué de trois acteurs : le CNRM (Centre National de Recherche Météorologique), le LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) et l'ISAE (Institut Supérieur de l'Aéronautique et de l'Espace) avec l'ONERA (Office National d'Études et de Recherches Aérospatiales).

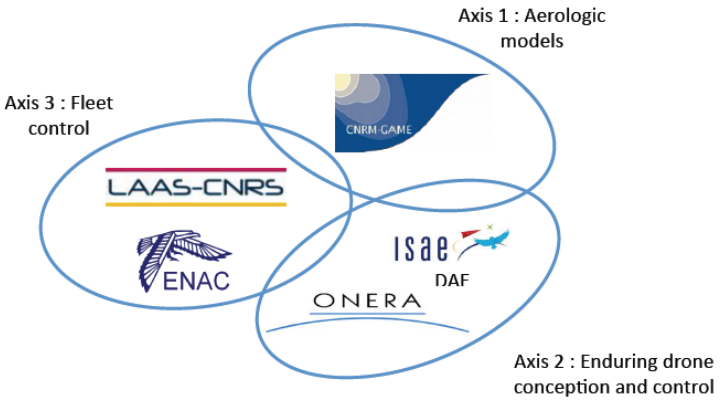


FIGURE 4.1 – Les partenaires du projet *Skyscanner*

Notre équipe est comprise au sein du groupe composé de l'ISAE et de l'ONERA (*Axis 2*) et chargé de la conception et le contrôle d'une drone à longue endurance. Le travail de l'ISAE-*Skyscanner* répond donc aux deux missions complémentaires suivantes :

1. Le besoin d'un drone d'observation et d'analyse des nuages manifesté par le CNRM (Centre National de Recherche Météorologique)
2. Le besoin d'un programme de design de drone manifesté par l'ONERA, à travers notre tuteur Emmanuel BÉNARD

Dans cette optique, notre travail a consisté à développer un outil informatique de design de drone répondant aux attentes de M. BÉNARD, et à l'utiliser pour développer un drone répondant aux attentes du CNRM. Pour M. BÉNARD le programme doit :

— Intégrer les différentes disciplines nécessaires au design du drone

— Intégrer un critère d'optimisation et un ensemble de contraintes de design associés à ces disciplines
— Réaliser une optimisation sur un ensemble de paramètres décrivant l'appareil et renvoyer un aéronef optimisé

Pour le CNRM le drone doit :

— Être capable de réaliser les missions spécifiées (au nombre de 3) avec un critère de performance (temps de mesure) à maximiser
— Respecter certaines contraintes supplémentaires (limitations liées à l'opérabilité du drone

Attentes Nous avons identifié 3 principaux axes de travail :

1. Développer les programmes traduisant les différentes disciplines intervenant dans le développement d'un drone (règles de dimensionnement, évaluation de performance)
2. Développer un optimiseur capable d'intégrer ces disciplines et de réaliser une optimisation sur un jeu de paramètres
3. Préciser les attentes du CNRM (cahier des charges du drone) et les formuler d'une manière à rendre possible leur intégration dans notre outil informatique

4.2 Constitution de l'équipe et partage du travail

L'équipe est constituée de 4 étudiants ingénieurs :

— Lauren Dargent
— Raphaël Julé
— Vincent Lefèvre
— Pierre Mermet-Bijon (chef du projet)

Cette équipe est encadrée par M. Emmanuel Bénard (enseignant-chercheur à l'ISAE) et M. El Khedim Bouhoubeiny (post-doc à l'ISAE). De plus, l'équipe dispose d'un encadrant pour la partie gestion de projet : M. Franck Gormand (*ALTRAN*).

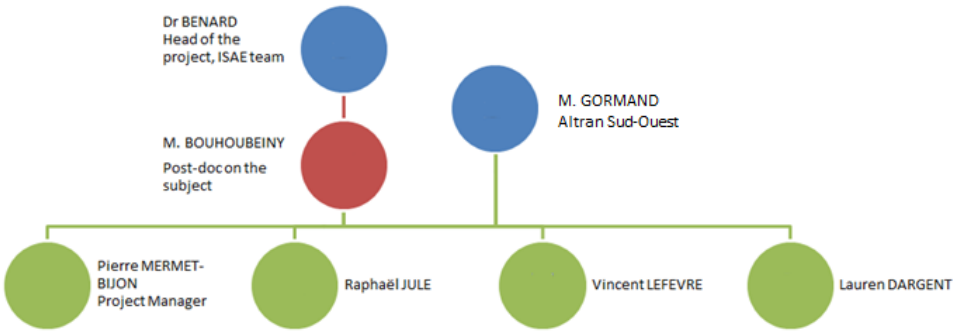


Figure 4.2 – L'équipe du projet

Répartition du travail Pour le début du projet, le travail a été réparti de la manière suivante : Lauren Dargent et Vincent Lefèvre se sont chargés de mieux définir le cahier des charges de la mission en partenariat avec le CNRM. Raphaël Julé et Pierre Mermet-Bijon se sont penchés sur le code afin de le mettre à jour vers des versions supérieures. Dans la suite du projet, les deux groupes se sont rassemblés pour améliorer ensemble le programme existant afin de développer un optimiseur répondant aux objectifs du projet.

4.3 Ressources

4.3.1 Disponibilités

Afin bien débuter le projet et de partir dans la bonne direction, des réunions hebdomadaires furent organisées avec les tuteurs directs du projet (M. Emmanuel Bénard et M. El Khedim Bouhoubeiny).

Ces réunions permettaient aux tuteur de donner des recommandations et de se tenir à jour sur le travail accompli.

Dans la suite du projet, ces réunions furent plus espacées car cela permettaient de dégager du temps de travail sur les horaires allouées au projet et la charge de travail extérieure était plus élevée. Le rythme de deux réunions par mois fut alors adopté. Ces réunions étaient plus destinées à avancer sur la programmation du logiciel d'optimisation.

Nous avons eu de très bonnes ressources au niveau de la disponibilité des personnes de notre équipe ainsi que celle de nos tuteurs. De plus, nous avons pu rapidement rencontrer un responsable du CNRM courant décembre afin de travailler sur la définition du cahier des charges de la mission, ce qui nous a permis d'avancer rapidement.

4.3.2 Supports de travail

Durant la première phase, les tâches étaient bien séparées, donc il n'y a pas eu de problème de ressources dans la gestion des différentes versions du fichier. Pour la deuxième phase (optimisation), les tâches sont devenues linéaires (nous devions absolument améliorer l'optimisation avant de pouvoir travailler à nouveau séparément les différents blocs). A ce moment-là, les quatre membres du groupe projet, rejoints par M. BOUHOUBEINY, ont travaillé sur les fichiers en parallèle, ce qui a engendré quelques problèmes de versions. Ces problèmes furent corrigés par la suite grâce à l'utilisation d'un utilitaire de partage de fichiers et de gestion des versions en ligne.

4.4 Découpage en tâches

Les trois principaux axes de travail étant été identifiés dans la section 4.1, ils ont été par la suite répartis en trois tâches distinctes :

- ***Tâche I*** : Définition de la mission
- ***Tâche II*** : Amélioration du logiciel d'optimisation
- ***Tâche III*** : Validation du logiciel et design du drone

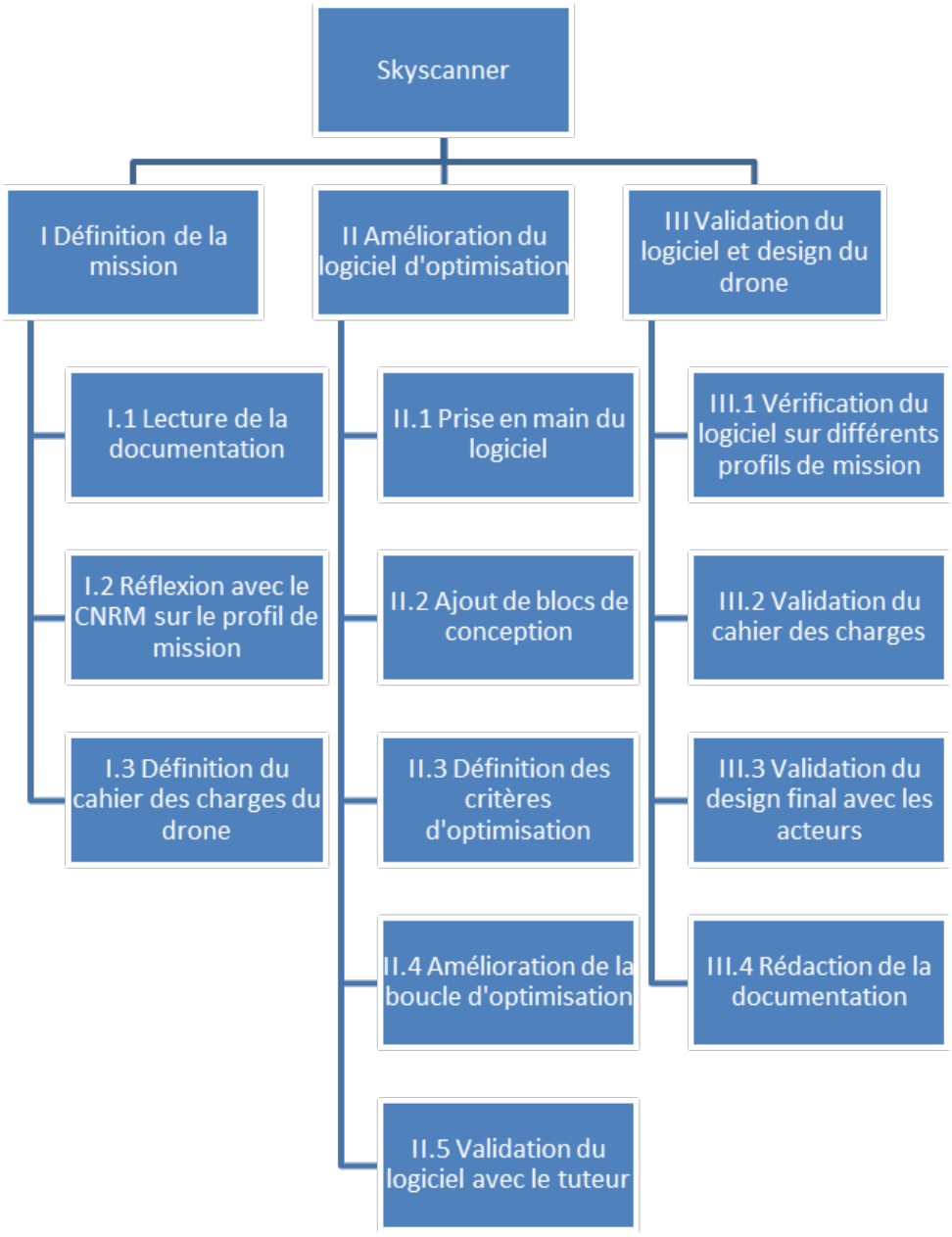


FIGURE 4.3 – Arbre des tâches

Les tâches sont alors divisées en sous-tâches qui sont visibles dans l'arbre des tâches. Cette arbre n'a subit qu'une seule modification depuis le début : la tâche *Ajout de blocs de conception* a été transformée en *Adaptation et ajout de blocs de conception*. En effet, nous ne sommes partis d'un code déjà existant et développé par l'un de nos tuteur : M. BOUHOUBEINY. Ce code étant écrit dans une ancienne version du langage *Python* et de l'utilitaire d'optimisation *OpenMDAO*, il a fallu le mettre à jour ce qui a engendré un délai supplémentaire.

Comme précisé dans la section 4.2, deux équipes travaillaient parallèlement lors de la première phase du projet. Lauren et Vincent s'occupaient de la définition de la mission (*Tâche I*) tandis que Pierre et Raphaël travaillaient sur la prise en main du logiciel, son adaptation et l'ajout de blocs de conception (*Tâche II.1* et *Tâche II.2*).

La *Tâche I* ayant pris fin fin décembre, l'équipe composée par Lauren et Vincent a rejoint l'équipe de Pierre et de Raphaël à partir de la *tâche II.3*. Le travail à partir de la *tâche II.3*, et jusqu'à la fin, concerne les quatre acteurs, dans le sens où l'enchaînement des tâches est linéaire, et le travail ne peut être avancé parallèlement. Cependant, chacun des membres de l'équipe était spécialisé dans une partie du code. Par exemple, Lauren pour le bloc *masse*, Pierre pour le bloc *aérodynamique*, Raphaël pour l'optimisation dans son ensemble et Vincent pour la validation du code.

4.5 Planning et déviations

La première phase du projet (*Tâche I*) qui se terminait avec la définition de la mission a bien été respectée dans les délais. En effet, une rencontre avec le CRNM début décembre nous a permis d'achever la définition du cahier des charges du drone avant la fin de l'année.

Retards En revanche, le travail sur le logiciel a dû être étalé dans le temps : la prise en main de l'utilitaire *OpenMDAO* fut progressive avec l'amélioration et la conception de blocs seuls puis assemblage de ces derniers. De plus, il a fallu mettre à jour cet utilitaire *OpenMDAO* ainsi que le langage utilisé *Python* ce qui n'avait pas été anticipé. Pour ces raisons, le début développement de la boucle d'optimisation a été retardé. Nous avions là aussi sous-estimé la durée nécessaire de cette étape. Mais un gros volume d'heure de travail a permis d'éviter d'accumuler du retard supplémentaire sur le planning. Le détail des heures de travail est visible dans la section 4.6.2.

Du fait de ce retard, les phases de validation du programme ont été décalé d'une quinzaine de jours environ. Le planning initial anticipait un possible retard : la fin du projet était prévue début mars. Ce retard de deux semaines fut donc sans conséquences pour la suite du projet et la livraison des livrables.

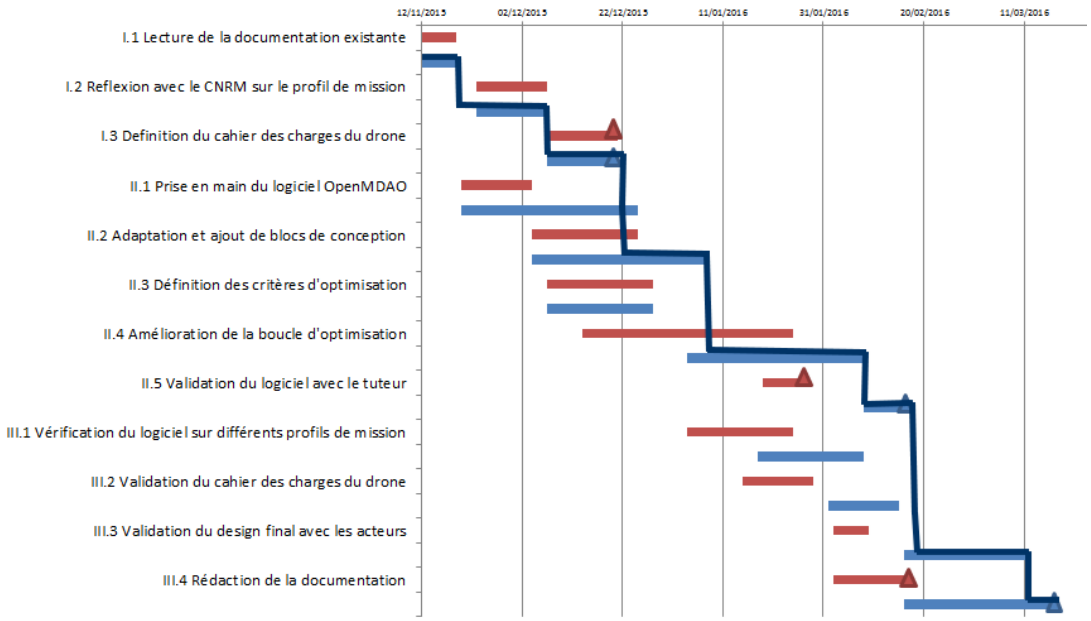


FIGURE 4.4 – Planning du projet avec le chemin critique

Le chemin critique regroupe les étapes suivantes :

- ***Tâche I*** : étapes I.1 (lecture de la documentation), I.2 (réflexion avec le CRNM) et I .3 (définition du cahier des charges)
- ***Tâche II*** : étapes II.2 (adaptation et ajout des blocs de conception), II.4 (amélioration de la boucle d'optimisation) et II.5 (validation du logiciel avec le tuteur)
- ***Tâche III*** : étape III.3 (validation du design avec les acteurs) et III.4 (rédaction de la documentation)

Ce chemin critique a été rallongé de 15 jours par rapport au planning initial.

4.6 Coûts du projet

4.6.1 Coûts financiers et matériels

Dans le cadre du projet, nous travaillons sur des fichiers en langage *Python* en utilisant du code développé par M. Bouhoubeiny et un optimiseur open-source : *OpenMDAO*. Les machines utilisées sont nos ordinateurs personnels. Les déplacements réalisés sur les sites du CNRM et de l'ONERA n'ont engendré aucun coût financier particulier. Nous n'enregistrons donc aucun coût financier ou matériel concernant ce projet.

4.6.2 Coûts horaire et charge de travail effectué

Le nombre d'heures attribué au total pour ce projet dans notre formation est de 80h/personne, réparties sur 4 mois. Cependant, ce projet est désigné pour un minimum de 5 personnes. N'étant que 4, nous avons du fournir un plus grand nombre d'heures de travail détaillées dans le tableau suivant.

Tâche	Détail		Personne en charge	Temps de travail prévu (h. par personne)	Temps de travail réalisé (h. par personne)
Tâche I (Début novembre – Fin décembre)	Définition de la mission		Vincent LEFEVRE Lauren DARGENT	15	**18**
Tâche II (Mi-novembre – Mi-février)	Amélioration du programme d'optimisation	Mise à jour du code (*Python* et *OpenMDAO*)	Pierre MERMET-BIJON Raphaël JULE	10	**18**
		Développement de l'optimiseur	Pierre MERMET-BIJON Vincent LEFEVRE Raphaël JULE Lauren DARGENT	40	**54**
Tâche III (Mi-janvier – Fin mars)	Validation du programme d'optimisation		Pierre MERMET-BIJON Vincent LEFEVRE Raphaël JULE Lauren DARGENT	22	**32**
	Rédaction de la documentation			6	**7**

Figure 4.5 – Charge de travail

Charge de travail prévue par personne : 91h.
Charge de travail finale par personne : 111h.
Charge de travail supplémentaire par personne : 20h.

Les heures allouées pour le projet n'ont pas été suffisantes. Des séances de travail supplémentaires ont dû être instaurées notamment pour travailler sur la mise à jour du code, développer la boucle d'optimisation et valider le programme. La charge de travail supplémentaire est d'environ 20h/personne. Les tâches qui ont nécessités un plus grand nombres d'heures de travail que prévu sont visibles en rouge sur la figure 4.5.

4.7 Analyse des risques et plan d'action de maîtrise des risques

Risques	Conséquences	Actions préventives	Action effectuée
Problème de responsabilité (définition des domaines de responsabilité des acteurs)	Profil de mission mal défini, tension	Fonctionner par proposition de solution/correction	Oui
Non prise en compte d'éléments particuliers sur le drone (fabrication, coût...)	Drone non réalisable	Contact régulier avec l'encadrement,	Oui
Non prise en compte d'éléments extérieurs	Drone non opérationnel	Contact avec les autres acteurs (ENAC, CNRM), et documentation	Oui
Non prise en compte de la réglementation	Pas d'autorisation de voler pour le drone	Documentation sur la réglementation des drones (DGAC...)	Non
Mission et organisation encore peu claire	Gestion du temps hasardeuse	Revues d'avancement régulières Proposition itératives avec les acteurs	Oui
Manque de maîtrise de l'environnement logiciel	Difficultés et retards dus au changement de version d'OpenMDAO	Consulter le site internet d'OpenMDAO, effectuer le changement de version avant tout travail	Oui
Gestion des fichiers python	Ne pas tenir compte de la dernière version du fichier	Renvoyer le fichier à tout le groupe dès qu'il y a une modification, en notifiant la date et la modification. Ou utiliser un logiciel de gestion de version (type svn).	Oui

Risque annulé

FIGURE 4.6 – Analyse des risques

Le risque de la non prise en compte de la réglementation a été annulé. Ce risque avait pour conséquence une interdiction de vol des drones par la DGAC (Direction Générale de l'Aviation Civile). Cependant, le CNRM assure que les drones voleront dans des espaces aériens dédiés et protégés qui dispose d'un statut particulier. La réglementation en vigueur n'étant pas valable dans ce type d'espaces aériens, il n'y a plus lieu de tenir compte de ce risque.

Par rapport à l'analyse des risques initiale réalisé en début de projet, un seul risque a été ajouté. Il s'agit de la gestion des versions des fichiers *Python*. En effet, avec 4 membres de l'équipe travaillant sur le code en parallèle, une bonne coordination est nécessaire. Ce problème fut traité par la suite avec l'utilisation d'un utilitaire de partage des fichiers.

4.8 Revue des actions et des jalons

Les actions et les jalons sont résumés dans le tableau suivant. Les actions correspondent aux tâches définies précédemment sur la figure 4.3. Les jalons indiqués ici sont essentiellement les grandes réunions durant lesquelles le travail réalisé a été présenté.

Actions	Jalons	Date début de l'action	Date du jalon
	Réunion de lancement	05/11/2015	
I.1 Lecture de la documentation		12/11/2015	
II.1 Prise en main du logiciel OpenMDAO		18/11/2015	
I.2 Réflexion avec le CNRM sur le profil de mission		23/11/2015	
	Réunion revue de projet 1	25/11/2015	
II.2 Adaptation et ajout de blocs de conception		03/12/2015	
I.3 Définition du cahier des charges du drone		05/12/2015	
II.3 Définition des critères d'optimisation		07/12/2015	
	Mission du drone fixée	31/12/2015	
II.4 Amélioration de la boucle d'optimisation		04/01/2016	
III.1 Vérification du logiciel sur différents profils de mission		15/01/2016	
III.2 Validation du cahier des charges du drone		02/02/2016	
	Mid-Term technique	03/02/2016	
	Réunion revue de projet 2	03/02/2016	
II.5 Validation du logiciel avec le tuteur		09/02/2016	
III.3 Validation du design final avec les acteurs		17/02/2016	
III.4 Rédaction de la documentation		17/02/2016	
	Logiciel d'optimisation fixé	01/03/2016	
	Soutenance finale	04/03/2016	

FIGURE 4.7 – Revue des actions et des principaux jalons

4.9 Revue des livrables

Les livrable sont composés de trois blocs distincts. Il est à noter que la réalisation complète de certains livrables est dépendante de celle des livrables précédents. En effet, la conception du logiciel d'optimisation est partiellement dépendante de la définition du cahier des charges du drone. De même, la rédaction de la documentation est dépendante des deux autres livrables.

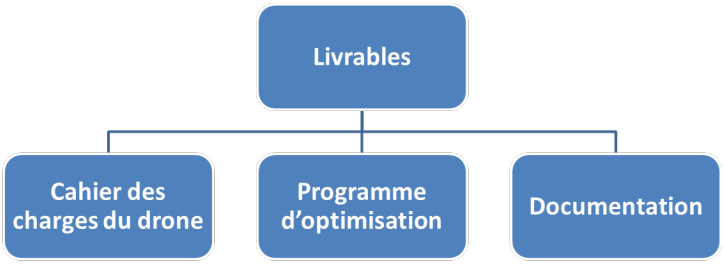


FIGURE 4.8 – Livrables

Cahier des charges du drone	
Responsables	Lauren DARGENT Vincent LEFEVRE
Description	Définition précise des missions des drones. Définition des paramètres atmosphérique associés à chaque mission et des paramètres géométriques associés à chaque drone. Segmentation des missions en cas de vol de façon à pouvoir les implémenter dans le logiciel d'optimisation.
Format du livrable	**Fichiers Excel** Entrée : paramètres de missions fixes et variables Sortie : cas de vol à implémenter dans le logiciel d'optimisation
Statut global du livrable	**Terminé**

FIGURE 4.9 – Livrable 1 : cahier des charges du drone

Logiciel d'optimisation	
Responsables	Lauren DARGENT Raphaël JULE Vincent LEFEVRE Pierre MERMET-BIJON
Description	Logiciel optimisant la géométrie d'un drone pour accomplir une mission donnée. La mission est segmentée en cas de vol. Le logiciel permet de fixer certains paramètres géométriques et d'en faire varier d'autres. Logiciel composé de plusieurs blocs liés les uns aux autres. Utilisation de OpenMDAO pour optimiser les paramètres géométriques du drone.
Format du livrable	**Fichiers Python** Entrée : paramètres du drone à faire varier et missions à accomplir Sortie : géométrie optimale du drone
Statut global du livrable	**Terminé**

FIGURE 4.10 – Livrable 2 : programme d'optimisation

Documentation	
Responsables	Lauren DARGENT Raphaël JULE Vincent LEFEVRE Pierre MERMET-BIJON
Description	Documentation explicative sur : - Le cahier des charges - Le logiciel d'optimisation
Format du livrable	**Documents PDF**
Statut global du livrable	**Terminé**

FIGURE 4.11 – Livrable 3 : documentation

4.10 Confrontation des résultats avec les objectifs initiaux

4.10.1 Qualité et contenu des livrables

En terme de qualité des livrables, les objectifs initiaux ont été respectés. Les livrables répondent au besoins exprimés dans le cahier des charges et par les tuteurs du projet. Des améliorations sont possibles dans le livrable associé au logiciel d'optimisation (figure 4.10) mais il a été décidé d'aller à l'essentiel vers les objectifs primaires plutôt que de se disperser afin de respecter les contraintes de temps.

4.10.2 Coût

Aucune dérive de coût n'a été enregistré. Cependant, une dérive considérable en terme de temps de travail a été remarquée. Comme précisé dans la section 4.6.2, le temps de travail supplémentaire par rapport à celui définit dans le planning au début du projet s'élève à 20 heures par personne. Ceci équivaut à une charge de travail finale supérieure de 22% à celle estimée au départ.

Conclusion

Ce projet nous a permis de poser les bases de l'architecture d'un logiciel d'optimisation de drone. Nous avons aussi pu montrer qu'il était pertinent d'utiliser un tel logiciel pour développer un aéronef. Au delà de ces succès, nos travaux ont aussi permis de cerner les difficultés et les limites liés au développement et à l'utilisation de ce genre de programme.

Le présent document a repris les différents éléments d'un programme d'optimisation et tout au long du rapport nous avons émis nos recommandations quand à la poursuite du développement de ces éléments. La structure de code que nous avons mis en place permet de progresser de manière indépendante sur ces différents axes pour faire gagner en maturité le programme et l'adapter aux situations étudiées.

Concernant la partie gestion de projet, il n'y a pas eu de modifications majeures du plan initial au cours du projet en dehors d'une légère révision du planning. En effet, suite à des contraintes non-anticipées comme l'adaptation du code à des versions plus récentes de *Python* et de *OpenMDAO*, le planning initial a du être revu pour absorber ce contretemps. Un risque de gestion des versions de code a aussi été clairement identifié et traité de notre côté, ce risque allant croissant (tant en criticité qu'en probabilité) avec la complexité du programme, nous conseillons fortement à une équipe qui reprendrait le projet de mettre en place une gestion rigoureuse des versions avant de commencer à travailler sur le code.

Pour finir, nous affirmons notre satisfaction quant au déroulement du projet. La gestion de ce dernier s'est faite de manière fluide, principalement grâce à l'investissement des différent membres de l'équipe et à l'efficacité de notre collaboration. A tous points de vue, ce projet a constitué une expérience très agréable et enrichissante pour chacun de nous.

Bibliographie

[1] V. Bonnin. *From Albetrosses to Long Range UAV Flight by Dynamic Soaring.* PhD thesis, ISAE, 2016.

[2] L. Pomarat C. Combier, M. Muller. Projet corsica : Pôle propulsion, conception et fabrication, 2015.

[3] B. Murat. *A Contribution to the Design of Long Endurance Mini Unmanned Aerial Vehicles.* PhD thesis, ISAE, 2012.