Projet d'Ingénieurie Simultanée

Projet Simulateur ERT

Henri Faure
Paul Germann
Jules Triomphe

Prof. Colin Jones
Dr. Philippe Müllhaupt

Remerciements

Nous aimerions remercier Dr. Müllhaupt P. pour son écoute et son aide hebdomadaire tout au long de ce semestre. De plus, nous remercions Mr. Noca, pour son aide sur la partie supersonique.

Table des matières

Liste des symboles

ΔC_{Dmax}	Augmentation maximale de la traînée sur la région transsonique	[-]
ΔC_{dS}	Élévation de traînée supersonique pour un nombre Mach donné 'M'	[-]
ΔC_{dT}	Élévation de traînée transsonique pour un nombre Mach donné 'M'	[-]
$\frac{dT}{dh}$	Variation linéaire de la température	$[\mathrm{K\,km^{-1}}]$
γ	Coefficient adiabatique ou coefficient de Laplace	[-]
ν	Viscosité cinématique	$[\mathrm{ft^2}s^{-1}]$
ϕ	Taux d'humidité dans l'air	[-]
ρ_{as}	Densité de l'air sec	$[\mathrm{kg\,m^{-3}}]$
a	Vitesse du son	fps
C_D	Coefficient de traînée total	[-]
C_x	Coefficient de traînée	[-]
Cd_b	Coefficient de traînée de base	[-]
$Cd_{f,body}$	Coefficient de traînée pour le corps	[-]
$Cd_{f,fins}$	Coefficient de traînée pour les ailerons	[-]
$Cd_{f,pro}$	Coefficient de traînée pour les protubérances	[-]
Cd_F	Coefficient de traînée due à la friction	[-]
Cf, Cf_1, Cf_2	Coefficient de frottement de la surface compressible	[-]
$Cf^{'}, Cf_1^{'}, Cf_2^{'}$	Coefficient de frottement de la surface incompressible	[-]
$Cf_{ro}^{'}, Cf_{ro,1}^{'}, Cf_{ro,2}^{'}$	Coefficient de frottement de la surface incompressible avec rugosité	[-]
Cf_λ	Coefficient de frottement moyen pour chaque aileron	[-]
$Cf_{final}, Cf_{final,1}, Cf_{final,2}$	Coefficient de frottement de la surface final	[-]
Cf_{pro}	Coefficient de frottement de protubérance	[-]
$Cf_{ro}, Cf_{ro,1}, Cf_{ro,2}$	Coefficient de frottement de la surface compressible avec rugosité	[-]
g	Accélération gravitionnelle	$[\mathrm{m\,s^{-2}}]$
h	Altitude par rapport au niveau de la mer	[m]
h_{sol}	Altitude du sol d'envoi de la fusée	[m]
K_F	Facteur d'interférence réciproque des ailerons et de la patte de lancement avec le corps	[-]
L	Longueur totale de la fusée	[inches]

M_d	Nombre de Mach de divergence en transsonique	[-]
M_F	Nombre de Mach final dans la région transsonique	[-]
p_{o}	Pression au niveau de la mer	[Pa]
p_{ws}	Pression saturante de la vapeur d'eau	[Pa]
r	Constante des gaz parfaits	$[\mathrm{J\,kg^{-1}\,K^{-1}}]$
R, R_1, R_2	Nombre de Reynolds compressible	[-]
R_a	Constantes de gaz de la vapeur d'air	[-]
R_w	Constantes de gaz de la vapeur d'eau	[-]
S	Section de la source	[S]
T_{o}	Température au niveau de la mer	[K]
T_{sol}	Température au sol	[K]
v	Vitesse de la source	$[\mathrm{m\,s^{-1}}]$
x	Taux de vapeur d'eau dans l'air	[-]
x_s	Taux de vapeur d'eau à saturation	[-]

Chapitre 1

Introduction

L'EPFL Rocket Team (ERT) est une équipe d'une cinquantaine d'étudiants de l'EPFL, de l'HEIG-VD et de la ZHAW. Elle participe à la plus grande compétition estudiantine de lancement de fusées : l'*Intercollegiate Rocket Engineering Competition* (IREC) lors de la *Spaceport America Cup* (SA Cup). Cette compétition est sous-divisée en 2 catégories ayant 3 sous-catégories chacune. Les 2 catégories sont des objectifs d'altitude : 10 000 ou 30 000 pieds, soit 3048 m ou 9144 m. L'IREC est en effet une compétition de précision et 350 points sur 1000 sont accordés en fonction de l'altitude atteinte, selon une échelle linéaire avec la distance, jusqu'à $10\,000 \pm 3000$ pieds pour la première catégorie.

L'ERT participe dans la sous-catégorie propulsion solide avec un moteur acheté dans le commerce, dans la catégorie 10 000 pieds. Les moteurs achetés dans le commerce ayant une poussée garantie à $\pm 10\%$, il est nécessaire de propulser la fusée à une altitude plus élevée que les 10 000 pieds visés afin de pouvoir rectifier l'apogée à l'aide d'aérofreins et garantir une apogée dans une fenêtre centrée sur l'altitude visée.

Afin de corriger l'apogée, un simulateur MATLAB® développé par l'équipe produit une table à deux dimensions qui définit l'ouverture nécessaire des aérofreins en fonction de l'altitude et de la vitesse du lanceur. En plus de produire cette table d'aérofreins, ce simulateur utilise des équations connues et adaptées aux caractéristiques du lanceur. Il n'est toutefois pas encore optimal et peut être amélioré. Trois axes de développement principaux ont été identifiés et font l'objet du projet de Bachelor des auteurs de ce rapport :

- L'amélioration du modèle atmosphérique grâce aux données récoltées par les stations météorologiques des alentours du site de lancement à la Spaceport America au Nouveau-Mexique, USA,
- Le développement d'une interface graphique utilisateur afin de faciliter la prise en main du simulateur,
- La préparation du simulateur à un vol supersonique dans l'optique d'une participation dans la catégorie 30 000 pieds de la compétition.

Plusieurs développements annexes ont aussi été effectués afin d'augmenter les fonctionnalités du simulateur et approfondir les connaissances de l'équipe pour des vols supersoniques :

- Développement d'analyses de stabilité,
- Simulations sur Fluent de vols supersoniques.

Chapitre 2

Atmosphère Standard

2.1 Atmosphère Standard US

L'atmosphère standard ou atmosphère normalisée était utilisée sur le simulateur MATLAB®. Cette dernière définit des conditions normales de température et de pression (CNTP) valables seulement dans la troposphère, c'est-à-dire jusqu'à 10 km d'altitude. Les trois principales données, température, pression et densité varient en fonction de l'altitude pour un lieu géographique donné et sont d'une importance générale pour un grand nombre d'études scientifiques. Cependant, ces données ne sont pas adaptées au lieu dans lequel se déroule la SA Cup. Ainsi, il nous faut des données plus précises, un modèle plus optimal, adaptable à plusieurs environnements précis. Les expressions suivantes décrivent l'atmosphère normalisée [6] :

Température

- *La température évolue comme :*

$$T = T_o + \frac{dT}{dh}\frac{h}{1000} \tag{2.1}$$

avec $dT/dh = -6.5[°C/km]$

Pression

- *La pression évolue comme :*

$$p = p_o \cdot (1 + \frac{dT}{dh}\frac{h}{1000 \times T_o})^{\frac{-1000 \times g}{r \times dT/dh}} \tag{2.2}$$

Densité

- *La densité évolue comme :*

$$\rho = \rho_{as} \cdot \frac{1+x}{1 + x \times R_w/R_a} \tag{2.3}$$

où $\rho_{as} = \frac{p}{rT}$ provient des gazs parfaits. x est exprimé à l'aide du taux d'humidité dans l'air ϕ et du taux de vapeur d'eau à saturation x_s :

$$x = x_s \cdot \phi \tag{2.4}$$

Le taux de vapeur d'eau à saturation x_s peut s'exprimer comme :

$$x_s = \frac{0.62198 \times p_{ws}}{p_{atm} + p_{ws}} \tag{2.5}$$

Avec la pression saturante de la vapeur d'eau p_{ws} :

$$p_{ws} = \frac{e^{77.345+0.0057T-7235/T}}{T^{8.2}} \tag{2.6}$$

2.2 Radiosonde

Une radiosonde est un instrument de télémétrie généralement porté par le ballon, qui mesure divers paramètres atmosphériques et les transmet par radio à un récepteur au sol. Les radiosondes sont lancées généralement au dessus de station météorologiques ou d'aéroport privés ou publics. Grâce à des données récoltées au dessus de la ville de Santa-Teresa [5], au Nouveau-Mexique, nous avons pu analyser et comparer l'atmosphère expérimentale avec celui, théorique, implémenté dans le simulateur.

2.3 Comparaison

La station météorologique de Santa-Teresa se situe à une centaine de kilomètres à vol d'oiseau au sud du lieu d'envoi de la fusée : la Spaceport America. Le climat du Nouveau-Mexique est globalement aride, ainsi, on considère que les données de radiosonde récoltées sont fiables pour ce lieu.

PRES hPa	HGHT m	TEMP C	DWPT C	RELH %
1000.0	6			
925.0	715			
871.0	1252	37.0	3.0	12
862.0	1348	34.6	0.6	12
850.0	1477	33.0	1.0	13
838.0	1605	31.8	0.8	14
816.7	1829	29.6	0.5	15
788.7	2134	26.7	0.1	18
761.7	2438	23.7	-0.4	20
735.5	2743	20.8	-0.8	24
700.0	3175	16.6	-1.4	29
695.0	3236	16.0	-2.0	29
660.6	3658	11.8	-2.5	37
613.8	4267	5.7	-3.2	53
591.7	4572	2.6	-3.6	64
590.0	4596	2.4	-3.6	65
583.0	4693	4.4	-27.6	8
579.0	4749	4.6	-32.4	5
569.9	4877	4.0	-32.4	5
528.0	5495	0.8	-32.2	6
505.0	5851	-2.1	-23.1	18
500.0	5930	-2.7	-23.7	18
489.5	6096	-4.1	-24.8	18
470.8	6401	-6.5	-26.9	18
458.0	6616	-8.3	-28.3	18

FIGURE 2.1 – Données de radiosonde à Santa-Teresa : 20 juin 2018

Altitude au-dessus du niveau de la mer [m]	Température [°C]	Pression absolue [hPa]	Densité de l'air [kg m^{-3}]
-1000	21.50	1140	1.374
0	15.00	1013	1.225
1000	8.50	899	1.112
2000	2.00	795	1.007
3000	-4.50	700	0.909
4000	-11.00	616	0.819
5000	-17.50	540	0.736
6000	-24.00	472	0.660
7000	-30.50	410	0.590

TABLE 2.1 – Données du modèle atmosphérique standard

Nous pouvons remarquer que les valeurs diffèrent nettement pour chaque altitude. Cela semble évident car il faut le rappeler, les données standards sont une moyenne de nombreux résultats acquis partout sur le territoire américain. Au-delà, nous pouvons prendre en compte une hausse de température, dû au réchauffement climatique, depuis la création de cette atmosphère standard dans les années 70.

2.4 Correction

Le but est de comprendre les similarités et différences entre le modèle expérimental et théorique, et d'adapter ce dernier en conséquence.

2.4.1 La Température

La température expérimentale est représentée sur la figure suivante :

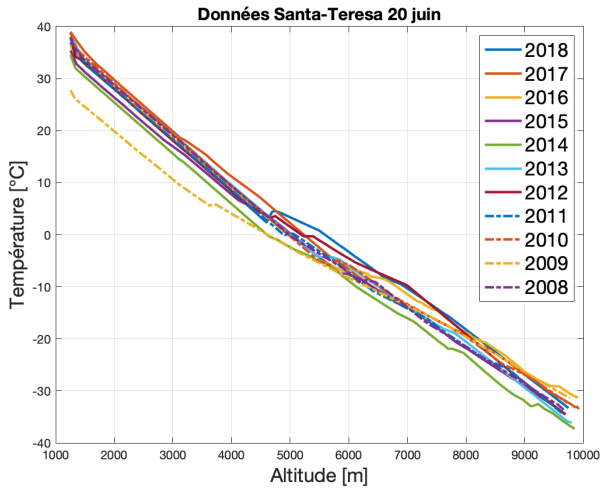


FIGURE 2.2 – Variation de la température en fonction de l'altitude

Tout d'abord, nous pouvons constater qu'elle varie linéairement entre 1000 m et 5000 m, comme pour le modèle standard. Visuellement et numériquement, nous pouvons calculer la pente qui est de -9.5 [°C/km]. Cette correction sera ensuite rapportée à l'expression de la température théorique (2.1). De plus, il faut s'assurer de lancer la fusée avec la

température exacte au sol. Pour cela, il suffit de décaler la courbe théorique du modèle atmosphérique normalisée vers la droite. Ce décalage correspond à l'altitude d'envoi de la fusée qui est de $h_{\text{sol}} = 1382\,\text{m}$ au Nouveau-Mexique. L'ensemble des modifications peuvent être résumées sur le graphique suivant :

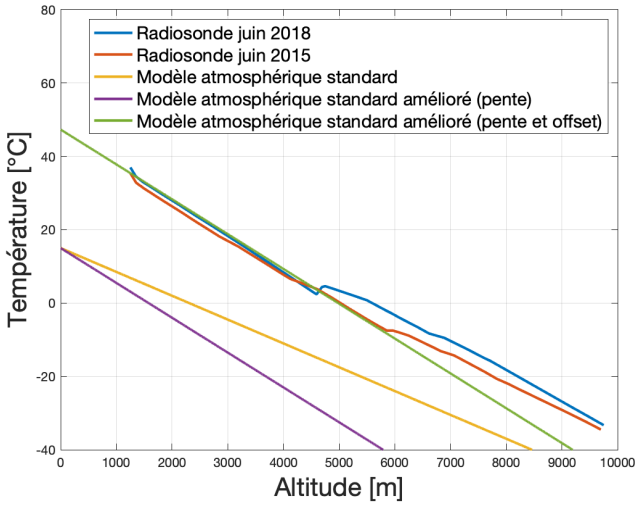


FIGURE 2.3 – Amélioration du modèle de la température standard

L'expression de la température devient alors :

$$T = T_{\text{sol}} + \frac{dT'}{dh}\frac{h - h_{\text{sol}}}{1000} \tag{2.7}$$

avec $\frac{dT'}{dh} = -9.5[°C/km]$

2.4.2 La Pression

La pression (2.2) définie à la section 2.1 dépend de la pression et de la température du niveau de la mer, puis de la pente de la température. En récupérant les modifications faites à la section 2.4.1, ainsi qu'en modifiant $T_o \Rightarrow T_{sol} \approx 35°C$ et $p_o \Rightarrow p_{sol} = 86000Pa$, nous obtenons la nouvelle expression de la pression standardisée améliorée :

$$p = p_{\text{sol}} \cdot (1 + \frac{dT'}{dh}\frac{h - h_{\text{sol}}}{1000 \times T_{\text{sol}}})^{\frac{-1000 \times g}{r \times dT/dh'}} \tag{2.8}$$

L'ensemble des modifications peuvent être résumées sur le graphique suivant :

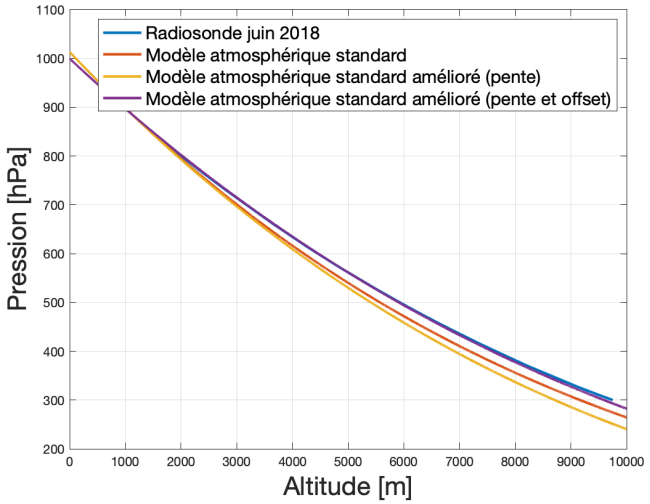


FIGURE 2.4 – Amélioration du modèle de la pression standard

2.4.3 La Densité

La densité (2.3) définie à la section 2.1 dépend de l'expression de la pression et de la température. L'expression de la densité standardisée améliorée prend la même forme que (2.3) mais en tenant compte des nouvelles expressions de la température (2.7) et la pression (2.8)

L'ensemble des modifications peuvent être résumées sur le graphique suivant :

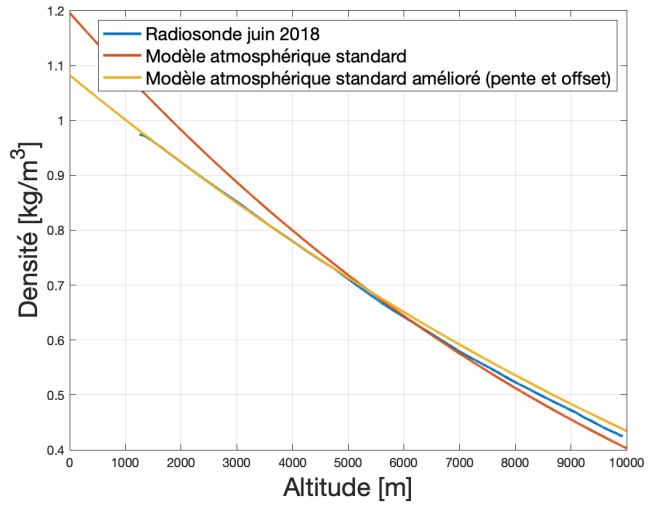


FIGURE 2.5 – Amélioration du modèle de la densité standard

Chapitre 3

Interface Graphique Utilisateur

3.1 Considérations

Plusieurs outils et langages de programmation se prêtent au développement d'interfaces graphiques. MATLAB®, bien que capable d'en faire, n'est pas optimisé pour ce type d'utilisation. Par ailleurs, c'est un logiciel payant, une limitation importante pour la distribution future du simulateur et la collaboration pour faire évoluer le code. Il a donc été décidé d'adopter un autre langage de programmation. Parmi les options disponibles, Python a été retenu pour ses nombreux modules adaptés à la création d'interfaces graphiques et aux calculs nécessaires pour effectuer la simulation de vol, sa documentation extensive, son caractère libre, sa large adoption et sa facilité d'apprentissage.

3.2 Opportunités

La traduction du simulateur de vol en Python a permis de restructurer le code afin de le rendre plus modulaire tout en formant les programmeurs à utiliser ce langage et les équations du simulateur.

3.3 Feuille de route

Le passage du langage MATLAB® au Python a nécessité plusieurs étapes :

1. Créer une définition plus modulaire de la fusée :
   (a) Définir une fusée comme une suite d'étages, chacun composé d'un corps et, si pertinent, un ou plusieurs moteurs et jeux d'ailerons,
   (b) Implémenter des classes pour chaque objet : Rocket, Stage, Body, Motor, Fins,
   (c) Implémenter les méthodes nécessaires dans chaque objet de la hiérarchie afin de pouvoir obtenir les valeurs requises dans le simulateur ;
2. Implémenter la fonction de traînée,
3. Choisir l'intégrateur adéquat pour le simulateur en une dimension (1D) et l'implémenter,
4. Tester les résultats du simulateur 1D et les comparer avec les résultats du code Matlab,
5. Concevoir le design de l'interface graphique utilisateur (GUI),
6. Écrire le code de la GUI,
7. Tester l'interface entre la GUI et le simulateur 1D.

Ces étapes ont été effectuées, mais elles ne sont pas entièrement finies, et il reste un certain nombre de points à améliorer afin de pouvoir, à minima, utiliser le simulateur implémenté en Python :

1. Adapter le code du simulateur 1D pour fournir les sorties voulues dans la GUI,
2. Implémenter la fonction d'état de la fusée en trois dimensions (3D),
3. Adapter les classes afin d'accommoder les besoins du simulateur 3D,
4. Comparer les résultats du simulateur 3D sur Matlab et sur Python,
5. Adapter la GUI au simulateur 3D.

D'autres améliorations pourront ensuite être implémentées, notamment :

- La possibilité de choisir le type de nose cone,
- La possibilité de choisir le type d'ailerons,
- Un simulateur capable de simuler correctement un vol en supersonique,
- Un simulateur capable de simuler correctement un vol trans-stratosphérique (i.e. étendre le modèle atmosphérique implémenté),
- Tester le simulateur avec des conditions non-standard et rajouter une gestion d'erreurs en conséquence.

3.4 Structure du code

Le code est basé sur le modèle de la fusée, dont les différentes parties sont implémentées en plusieurs classes en Python. L'idée principale du simulateur est de séparer la fusée en étages. Dans le monde du spatial, ceux-ci sont définis par les plans de séparation de la fusée. Pour l'ERT, ceux-ci sont définis par chaque interface ayant un coupleur ou un slide-in. La structure de chaque classe est décrite selon la Figure 3.1. Les termes sont en anglais car le code est destiné à être utilisé par des personnes non-francophones.

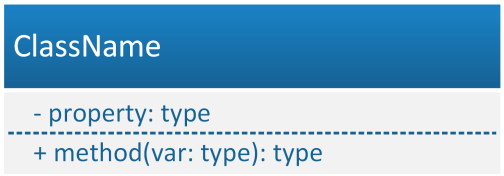


FIGURE 3.1 – Diagramme UML d'une classe

Chaque fusée a au moins un moteur. Ceux-ci peuvent être ajoutés à la fusée ou testés individuellement à l'aide de la classe *Motor* représentée dans la Figure 3.2. Les méthodes implémentées dans la classe permettent en effet d'obtenir les données de poussée, de masse et d'inertie du moteur considéré à tout instant t. Cette classe prend comme entrée uniquement le chemin d'accès au fichier du moteur, au format *.eng* ou *.txt*, qui peut être téléchargé depuis le site www.thrustcurve.org. Ces fichiers moteurs doivent être placés dans le dossier ***Motors***. Attention toutefois, car bien que la possibilité existe d'avoir plusieurs moteurs dans la fusée, l'implémentation du décalage temporel entre les différents moteurs reste à faire.

Chaque étage peut désormais techniquement avoir plusieurs jeux d'ailerons.
Toutes les propriétés de cette classe (cf. Figure 3.3) doivent être définies lors de l'initialisation.

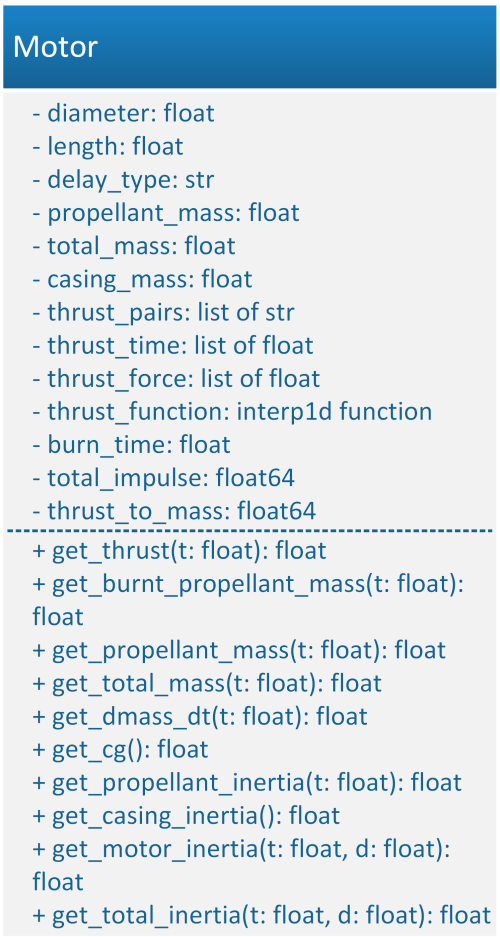


FIGURE 3.2 – Diagramme UML de la classe *Motor*

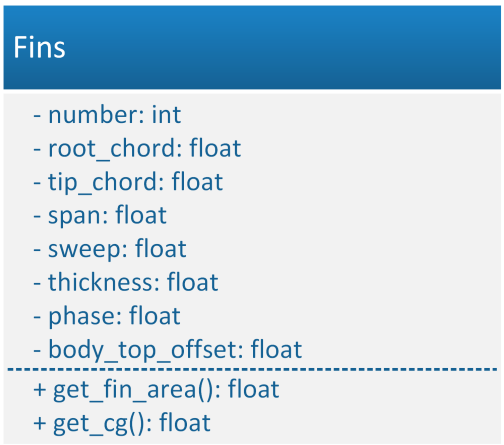


FIGURE 3.3 – Diagramme UML de la classe *Fins*

Enfin, chaque étage est composé d'un corps (*Body*, cf. Figure 3.4). Pour l'instant deux types de corps sont définis : une ogive tangentielle (pour le cône), et un corps cylindrique (pour chaque tube et transition régulière). Le type de corps, la position des diamètres du corps à chaque endroit de changement et la position de ces changements de diamètres doivent tous être définis lors de l'initialisation d'un corps ; les deux derniers paramètres doivent être rentrés dans des matrices NUMPY. Afin d'enrichir le simulateur, les différents types de nose cones ont été recherchés et cités dans la documentation de la classe. Ils

pourront être implémentés en temps voulu, lorsque de telles formes seront utilisées dans nos fusées.

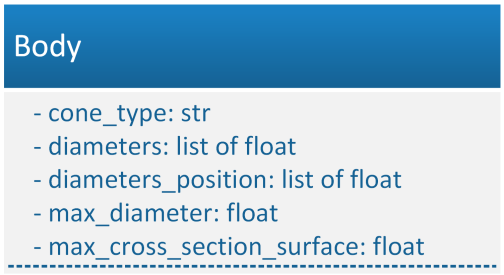


FIGURE 3.4 – Diagramme UML de la classe *Body*

Ces trois classes sont utilisées par la classe *Stage* (Figure 3.5). Toutefois, pour l'initialiser, seul un corps est nécessaire, ainsi qu'un nom, sa masse, son centre de masse et son inertie à vide (i.e. sans moteur ni ailerons). Ses méthodes *add_fins* et *add_motor*, dont les entrées sont les mêmes que celles nécessaires à l'initialisation des classes *Motor* et *Fins*, permettent en effet d'initialiser ces objets et de les rajouter à l'étage. Cette classe a également des méthodes afin d'obtenir la masse, la variation de masse et la poussée de l'étage à un moment donné en prenant en compte tous les moteurs de l'étage car cela est nécessaire pour déterminer l'équation du mouvement de la fusée.

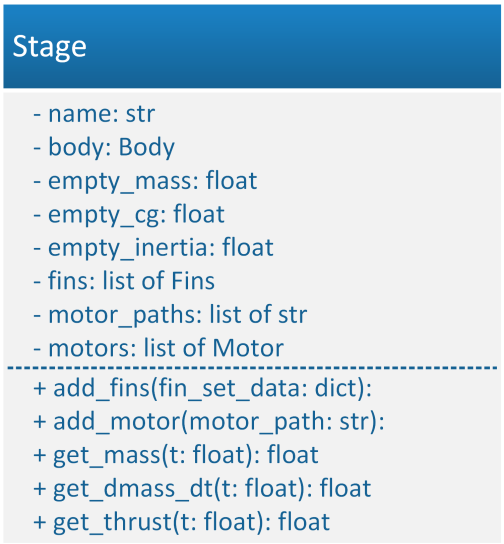


FIGURE 3.5 – Diagramme UML de la classe *Stage*

Chaque fusée est ensuite composée d'un ensemble d'étages. La classe *Rocket* (Figure 3.6) est initialisée avec une liste d'étages, qui peut être vide. Ceux-ci peuvent en effet être rajoutés après l'initialisation de la fusée en utilisant la méthode *add_stage*. Toutes les autres méthodes sont utilisées par la fonction de trainée ou par la fonction d'état de la fusée. Certaines propriétés de la fusée sont pour l'instant fixées car nous ne savons pas encore comment implémenter les aérofreins. Il faut en effet davantage caractériser leur effet afin de pouvoir les modéliser correctement dans le simulateur et les définir avec les valeurs importantes, à déterminer. Par ailleurs, la définition des aérofreins doit être propre à chaque modèle et nécessite une certaine modularité, particulièrement pour la fonction

de trainée de ceux-ci, qui doit encore être implémentée.

Cette classe a de multiples fonctions qui permettent d'aller chercher des informations sur le moteur ou les ailerons de la fusée assemblée. Toutefois, pour l'instant, la fonction de trainée *drag* et celles de la classe *Rocket* ne peuvent que prendre le premier jeu d'ailerons car davantage de recherches sont nécessaires afin de pouvoir caractériser l'influence, l'interférence et la pertinence de plusieurs jeux d'ailerons sur la fusée et comment modéliser celle-ci, puis l'implémenter.

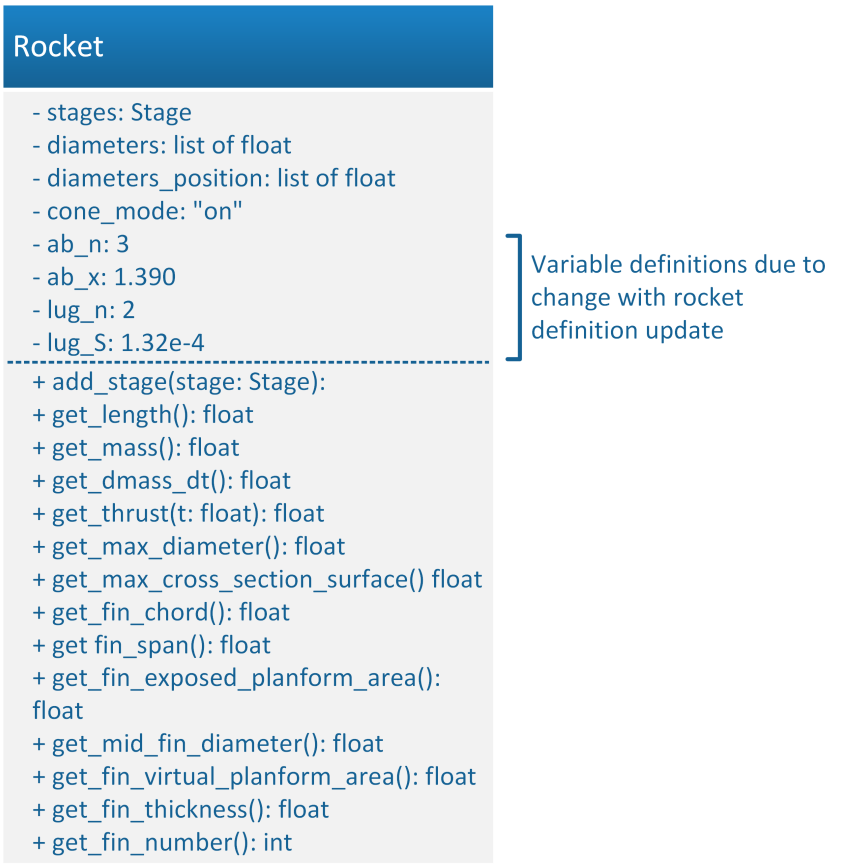


FIGURE 3.6 – Diagramme UML de la classe *Rocket*

La fusée est encapsulée dans un package *Rocket* avec ses quatre classes, représenté sur la Figure 3.7.

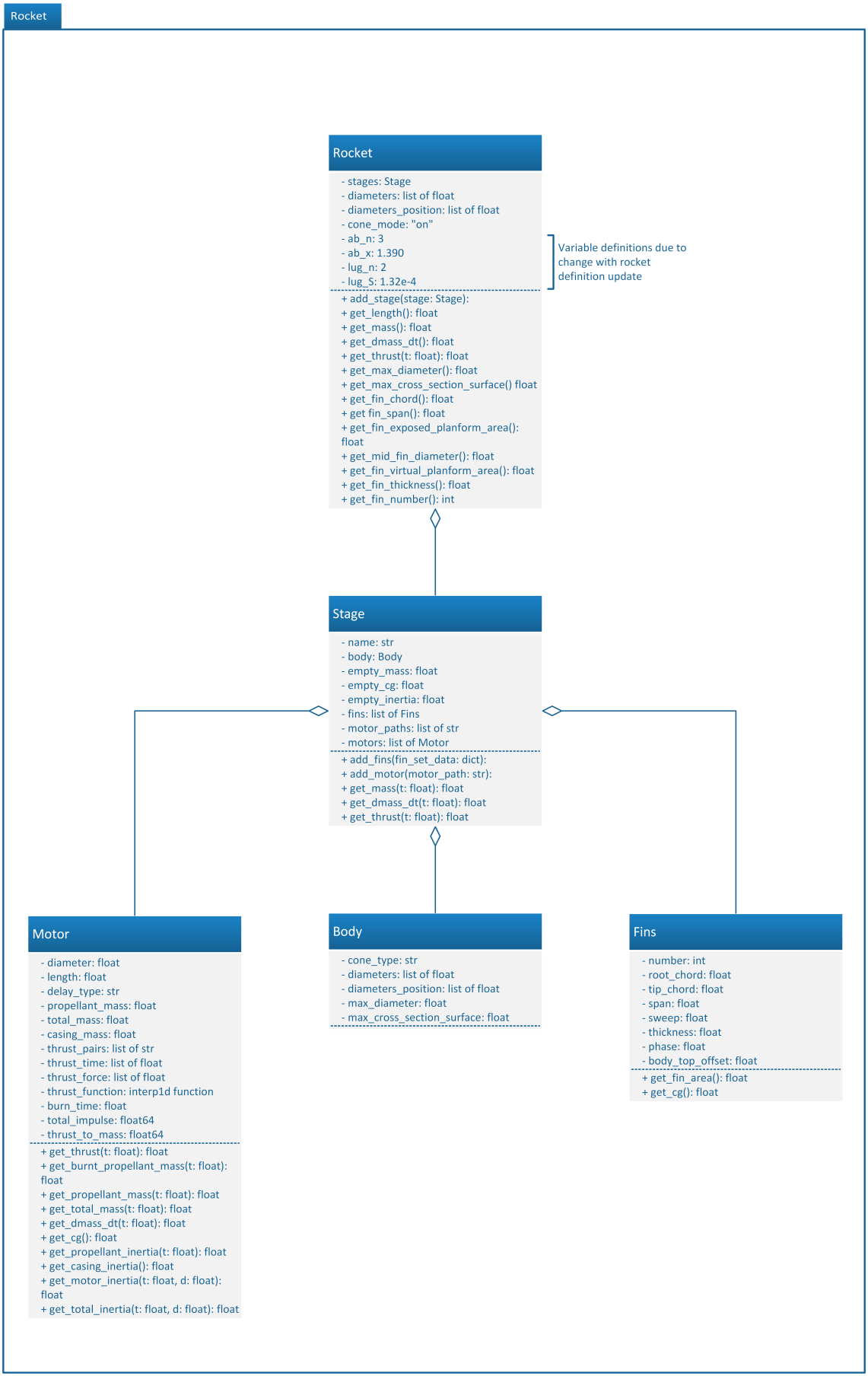


FIGURE 3.7 – Diagramme UML du package *Rocket*

Les paramètres variables qui interviennent sont non seulement les caractéristiques de la fusée, mais également celles de l'environnement. Celui-ci est défini par la classe *stdAtmosUS*, qui prend en entrée l'altitude du sol par rapport au niveau moyen de la mer, la température au sol (en Kelvins), la pression (en Pascal) et l'humidité (en pourcentage) au sol. A partir de ces valeurs, la température, la pression, la densité de l'air, la vitesse du son et la viscosité de l'air peuvent être calculées grâce aux méthodes présentées sur la Figure 3.8.

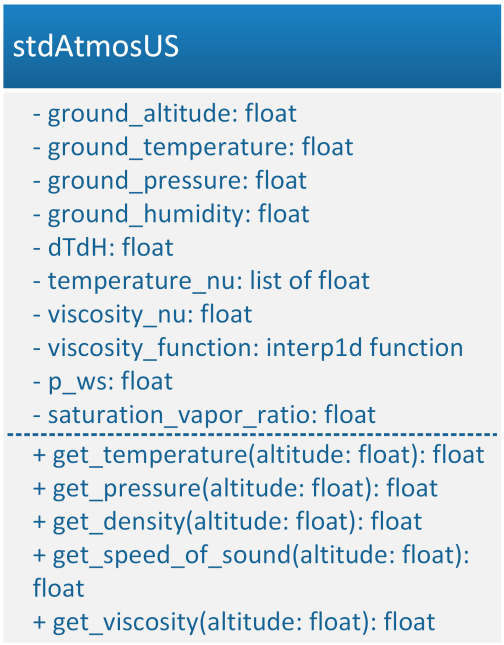


FIGURE 3.8 – Diagramme UML de la classe *stdAtmosUS*

Enfin, la simulation 1D de la trajectoire de la fusée peut être calculée à l'aide de la classe Simulator1D, qui prend en entrée un objet *Rocket* et un objet *stdAtmosUS*. Deux méthodes sont implémentées dans cette classe : la fonction *xdot* qui prend en entrée un temps t et un vecteur d'état x et retourne le vecteur d'état contenant la vitesse et l'accélération de la fusée au temps donné, et la fonction *get_integration* dont les entrées sont un nombre de pas d'intégration et le temps maximal jusqu'au quel l'intégration doit être faite.

L'intégrateur utilisé est la fonction *solve_ivp* de la librairie SCIPY.INTERPOLATE, dont les options sont semblables à celles de la fonction *ode45* de MATLAB®.

Plusieurs tests ont été réalisés afin de choisir cet intégrateur. Les résultats de celui-ci ont été comparés à ceux d'un autre intégrateur utilisant la méthode de Runge-Kutta du 5^e ordre, l'intégrateur *ode* avec l'option 'dopri5' de la librairie SCIPY.INTERPOLATE, ainsi qu'aux résultats de la fonction *ode45* de MATLAB®. Les résultats de cette comparaison, basée sur la fusée Matterhorn III qui a volé à Cernier le 23 mars 2019, mais avec les paramètres environnementaux du vol de la compétition 2018 aux Etats-Unis, sont présentés dans le Tableau 3.1. Les données atmosphériques du lancement à Cernier étant en effet très approximatives, nous avons préféré effectuer ces tests avec des données de conditions réelles certaines.

	MATLAB®	Python	
Intégrateur	*ode45*	*solve_ivp*	*ode*
Avec la fonction de traînée	2022.79 m	2088.58 m	2072.52 m
Écart relatif	-	3.25%	2.46%
Sans la fonction de traînée	2705.86 m	2790.88 m	2783.79 m
Écart relatif	-	3.14%	2.88%

TABLE 3.1 – Comparatif des résultats des différents intégrateurs

Les différences s'expliquent par les approximations faites dans la fonction *ode45* et un peu par les erreurs d'arrondi des deux logiciels, qui mènent à des erreurs par vecteur d'état à un moment donnée de l'ordre de $10e-8$; erreurs qui s'accumulent ensuite. Par ailleurs, les différences entre les deux fonctions en Python proviennent de la légère différence de fonctionnement de ceux-ci. En effet, toutes deux contrôlent l'erreur en faisant l'hypothèse que la méthode du 4^e ordre est correcte, mais choisissent les pas d'intégration en utilisant la méthode précise du 5^e ordre en extrapolant localement, mais *solve_ivp* est basée sur *ode* et l'améliore, replaçant peu à peu celle-ci parmi les utilisateurs de Python. C'est pourquoi nous avons choisi d'utiliser la fonction *solve_ivp* pour le simulateur en Python. Les résultats devront être comparés aux résultats de vols réels, mais cela devra être fait une fois que le simulateur 3D aura été implémenté afin de comparer des choses similaires.

La composition de la classe *Simulator1D* est représentée sur la Figure 3.9 et le schéma final du fonctionnement du simulateur est représenté sur la Figure 3.10.

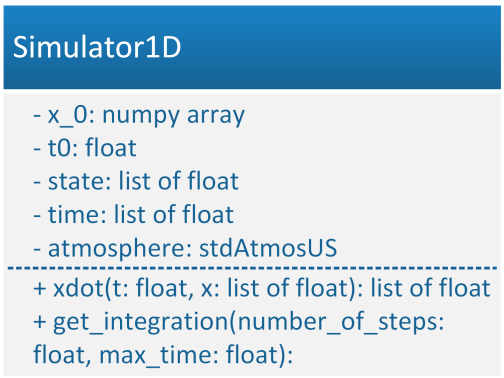


FIGURE 3.9 – Schéma UML de la classe *Simulator1D*

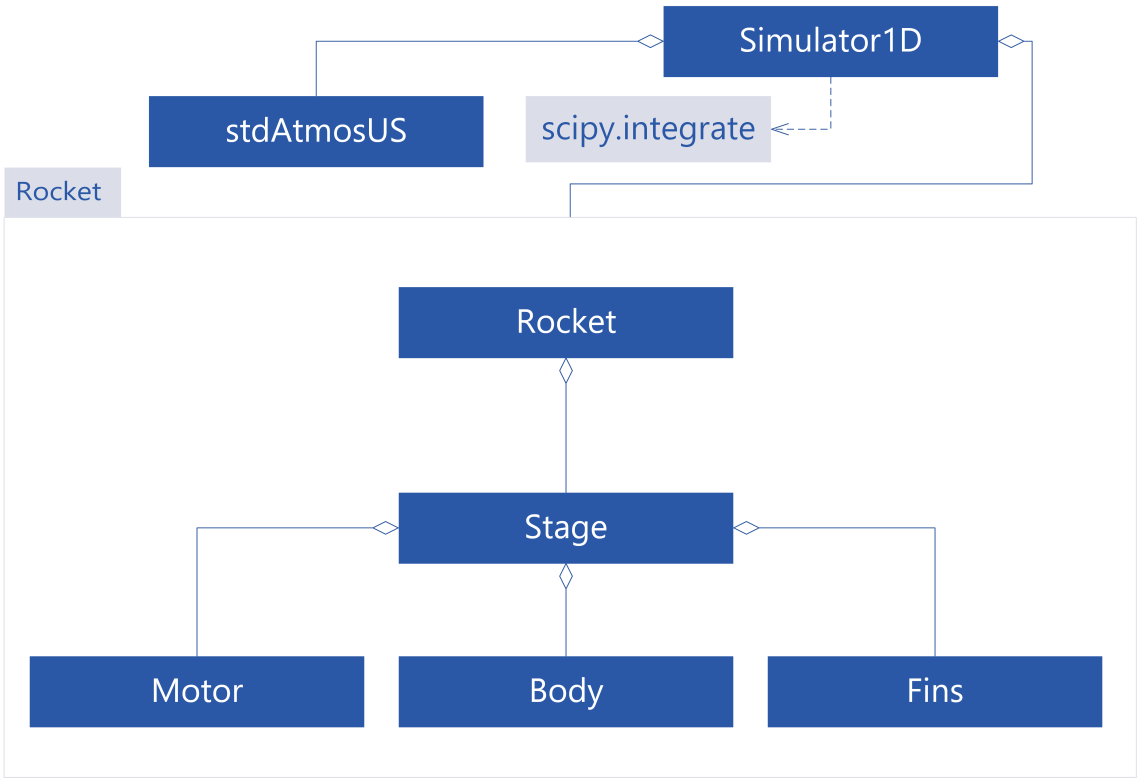


FIGURE 3.10 – Schéma UML du simulateur 1D

Afin de faciliter l'évolution du code, chaque fonction est documentée comme dans la Figure 3.11, et chaque classe est documentée de façon extensive au début du fichier l'incorporant. La fonctionnalité de chaque fonction est ainsi détaillée, ainsi que ses entrées et ses sorties.

```
def get_thrust(self, t: float) -> float:
    """
    Computes the current thrust, in [N].
    It is 0 if outside of burn time.

    :param t: time, in [s]
    :return: current thrust force, in [N]
    """
    if 0 <= t <= self.burn_time:
        return self.thrust_function(t)
    else:
        return 0
```

FIGURE 3.11 – Exemple de fonction commentée

Une fois ce code fonctionnel, nous avons pu le lier à l'interface graphique utilisateur (GUI), détaillée ci-après.

3.5 Code GUI

L'idée principale de cette GUI est de faciliter l'utilisation du simulateur par les membres du projet.

L'interface graphique du simulateur est inspirée de *OpenRocket* [9], le logiciel commercial de référence utilisé jusqu'à présent par la Rocket Team. L'intérêt principal de se séparer de *OpenRocket* [9] est de pouvoir implémenter à la GUI le simulateur de vol de la Rocket Team, afin de mieux maîtriser les équations physiques gouvernant la simulation. En particulier, les études météorologiques faites dans le cadre de ce projet permettent d'ajuster certains coefficients et de se rapprocher ainsi d'un modèle de vol le plus réaliste possible.

3.5.1 Tkinter, module de Python

L'interface graphique a été pensée sur Python. Ce langage est notamment doté de Tkinter. L'un des avantages de ce module est sa portabilité sur les OS les plus utilisés par le grand public. Il est dédié à la création de GUIs.

Ce module dispose d'une grande panoplie de "window gadgets", par contraction, widgets. Ceux-ci sont très nombreux, mais les principaux qui sont utilisés ici sont les suivants :

1. Button : Boutton poussoir
2. RadioButton : Boutton radio
3. Label : Etiquette
4. Entry : Champ de texte en entrée
5. Menu : Menu déroulant
6. Frame : Cadre
7. Scrollbar : Barre de défilement
8. Canvas : Conteneur d'objets graphiques 2D évolués
9. Treeview : Arborescence

3.5.2 Schéma GUI

La logique de la GUI est la suivante : la fenêtre principale est subdivisée en deux "frames". Une "frame" avec laquelle l'utilisateur peut directement interagir et une autre dans laquelle la fusée est dessinée en 2D, et où les données de la simulation sont chargées.

La première "frame" est elle-même subdivisé en quatre "frames" :

1. Arborescence : Cette fenêtre rend compte de l'extrême modularité de la fusée. Une arborescence est créée avec les différents étages de la fusée. Ceux-ci sont ensuite modulables grâce aux "buttons" "New Stage", "Remove", "Move up" et "Move down". A chaque nouvel étage créé, il est possible de leur attribuer un nom avec le "entry" disponible. Le "button" "Save" permet de valider la création de l'étage.
2. Corps/Moteur/Environnement : Cette fenêtre permet de sélectionner le corps que l'on souhaite intégrer dans l'étage sélectionné. Ainsi, les "buttons" "Nosecone", "Tube", "Fins", "Boat-Tail" sont disponibles. Dans cette "frame", il est également possible de choisir le moteur et de paramétrer l'environnement.
   En réalité, ce sont des "radiobuttons". Cela permet de faciliter l'utilisation de l'interface. En effet, des dossiers sont prédéfinis. Pour exemple, lorsqu'on clique sur le "radiobutton" "Nosecone", deux options s'offrent à l'utilisateur : soit de choisir le "nosecone" prédéfini de la fusée "Eiger", soit de "personalize" et de rentrer les paramètres qu'il souhaite.

3. Données : Cette fenêtre permet à l'utilisateur de renter les paramètres qu'il souhaite. Des "entry" sont disponibles à cet effet. Puis le "button" "Displays" ou "Save" permet d'enregistrer les paramètres entrés et d'afficher, s'il s'agit d'un corps, le corps dans le "canvas" prévu à cet effet.
4. Simulation : Cette dernière fenêtre ne comprend qu'un seul "button" : "Launch Simulation". Celui-ci a pour commande d'accéder au simulateur de la fusée. Il lance le simulateur en tenant compte de tous les paramètres entrés par l'utilisateur.

La deuxième "frame" permet d'une part d'afficher les caractéristiques de la fusée paramétrée par l'utilisateur, et d'autre part de dessiner la fusée dans un "canvas".
Chaque étage de la fusée est placé dans un "frame" indépendant, eux-mêmes subdivisés en "canvas" dans lesquels se trouvent les corps de la fusée.

3.5.3 Pistes d'améliorations de l'interface

Cette interface n'est pas encore aboutie. Une des premières nécessités est de rajouter une gestion des erreurs au code. Pour le moment, seule une exécution suivant parfaitement le tutoriel ci-dessous fonctionne. Il pourrait également être intéressant de pouvoir modéliser le ou les moteurs. En outre, il serait intéressant de pouvoir ajouter une représentation du centre de masse ainsi que du centre de pression afin de vérifier plus facilement certains critères indispensables de stabilité de la fusée. Enfin, il serait utile de rajouter certaines données qui ne sont pour le moment pas implémentées dans l'interface telles que, la masse de la fusée, sa vitesse maximale, son accélération maximale et sa stabilité. Pour plus d'informations, se référer directement à la documentation sur le code.

3.5.4 Tutoriel d'utilisation

1. Créer un nouvel étage :
   (a) Cliquer sur "New Stage"
   (b) Entrer le nom du nouvel étage dans la barre dédiée à cet effet
   (c) Cliquer sur "Save"
2. Intégrer les parties de la fusée dans l'étage créé précédemment : *(Attention à toujours bien veiller à ce que l'étage dans lequel vous importez un élément de la fusée soit sélectionné dans l'arborescence)*
   (a) Cliquer sur "Nosecone", "Tube", "Fins" ou "Boat-Tail" ; un menu déroulant s'affiche
   (b) Sélectionner "Eiger" pour prendre la partie correspondante de la fusée "Eiger"
   (c) Ou alors, sélectionner "Personalize" pour entrer d'autres paramètres, puis cliquer sur "Displays" pour afficher l'élément paramétré dans le cadre dédié à cet effet
3. Re-moduler la fusée : *(Attention à toujours bien veiller à ce que l'élément que vous souhaitez modifier soit sélectionné dans l'arborescence)*
   - Cliquer sur "Remove" pour supprimer une partie de la fusée
   - Cliquer sur "Move up" pour décaler un élément de la fusée vers la tête de celle-ci
   - Cliquer sur "Move down" pour décaler un élément de la fusée vers la queue de celle-ci
4. Lancer la simulation
   (a) Cliquer sur le bouton "Launch Simulation"
   (b) Les données de la simulation sont chargées dans le cadre dans lequel la fusée est esquissée

(c) Affichage des graphes en fenêtres externes

Ci-dessous, deux exemples d'exécution de la GUI, le premier modélisant Eiger et le deuxième modélisant Matterhorn III.

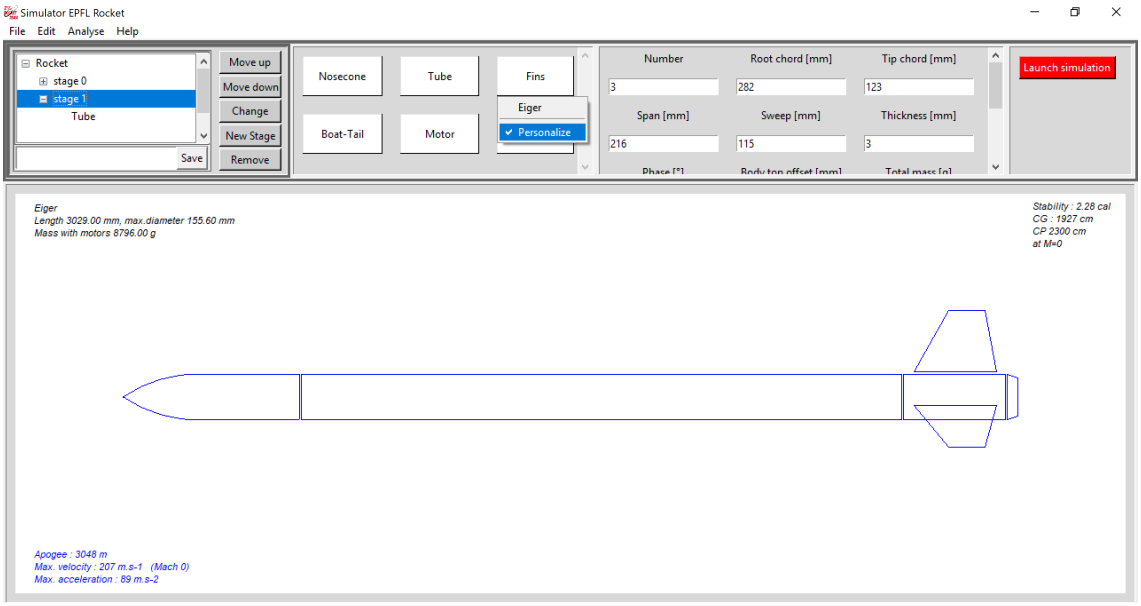


FIGURE 3.12 – Eiger

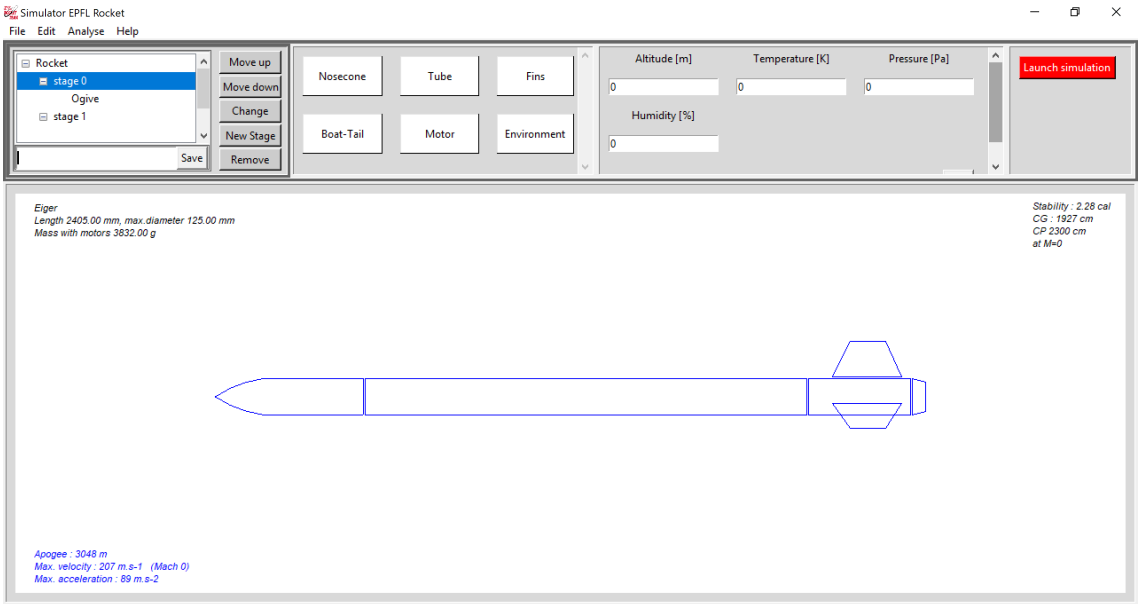


FIGURE 3.13 – Matterhorn III

Chapitre 4

Supersonique

Lorsqu'une fusée se déplace dans les airs, les molécules d'air à proximité de la fusée sont perturbées et se déplacent autour de celle-ci. Si la fusée passe à faible vitesse, la densité de l'air demeure constante (fluide incompressible). Mais pour des vitesses plus élevées, une partie de l'énergie de la fusée est transférée et va localement changer la densité de l'air. Cet effet de compressibilité modifie la force qui en résulte sur la fusée. L'effet devient plus important à mesure que la vitesse augmente. Près et au-delà de la vitesse du son, de petites perturbations de l'écoulement sont transmises à d'autres endroits de façon isentropique. Mais une forte perturbation génère une onde de choc qui affecte à la fois la portance et la traînée de la fusée.

4.1 Nombre de Mach

Pour expliquer chacun des phénomènes cités auparavant, nous introduisons le nombre de Mach. C'est un nombre adimensionnel permettant de savoir dans quel état d'écoulement on se situe (subsonique, transsonique ou supersonique).

$$M = \sqrt{\frac{Force\ d'inertie}{Force\ élastique}} = \frac{v}{a} = \frac{v}{\sqrt{\gamma r T}} \tag{4.1}$$

Avec v : vitesse de la source, a : vitesse du son, $\gamma = 1.4$ [-] et $r = 287$ [$\mathrm{J\,kg^{-1}\,K^{-1}}$]

$$\begin{cases} M < 0.7 & \text{Subsonique, la compressibilité peut être négligée} \\ 0.7 < M < 1.1 & \text{Transonique, apparition d'onde de choc sur la structure} \\ M > 1.1 & \text{Supersonique, onde de choc au deux extrémités de la fusée} \end{cases}$$

La figure 4.1 résume la répartition des pressions au passage des trois phases : subsonique, transsonique et supersonique. Lors du passage en transsonique, la répartition des pressions est sensiblement modifiée. En effet, l'apparition des ondes de choc et l'évolution de celles ci vont entraîner une dépression importance qui va modifier les valeurs de la traînée et de la portance.

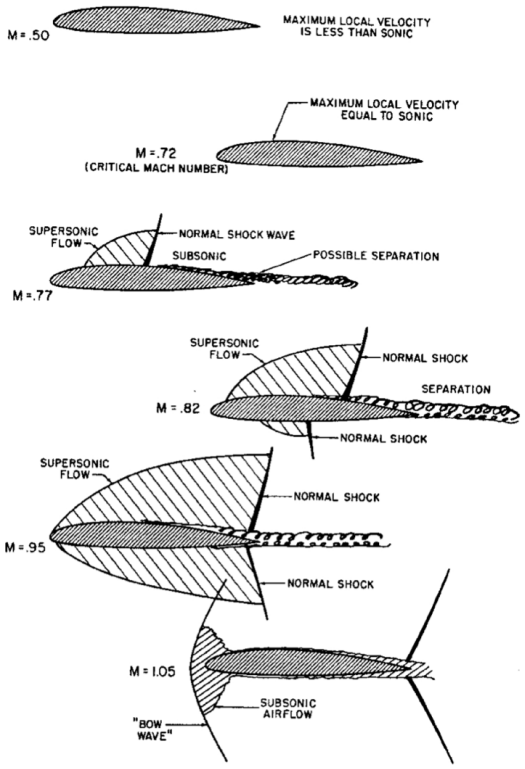
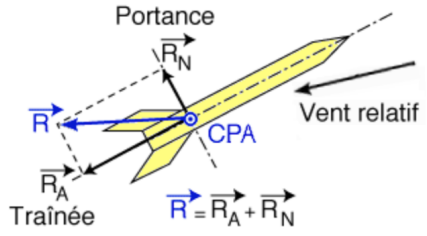


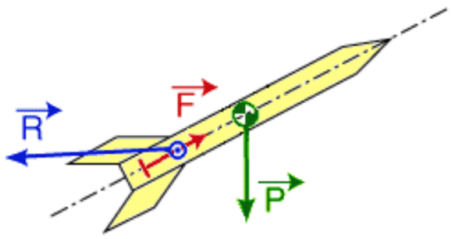
FIGURE 4.1 – Modèles d'écoulement transonique [7]

4.2 Prediction du Coefficient de Traînée

Pour ce problème, nous nous intéresserons au cas 1D de la fusée, c'est-à-dire, un angle d'incidence α nul. Cela signifie que la fusée vole face au vent. Cette dernière étant symétrique, il en résulte une portance nulle. Ainsi seul le résultat de la traînée influe sur la cinématique finale du problème.



(a) Décomposition en forces de la résistance du vent



(b) Décomposition de l'ensemble des forces résultantes lors du vol de fusée

FIGURE 4.2 – Forces résultantes sur une fusée [11]

Avec $\overrightarrow{R} = \overrightarrow{R_A}$ (cas 1D) et $\overrightarrow{F}$: Poussée du moteur, et $\overrightarrow{P}$: Poids de la fusée.

Connaissant les propriétés du moteur ainsi que le poids de la fusée, il nous reste à déterminer la force de traînée (4.2) pour appliquer la deuxième loi de Newton et trouver la trajectoire de la fusée.

La force de trainée à l'expression suivante :

$$F_x = \frac{1}{2}\rho v^2 S C_x = \frac{1}{2}\gamma p M^2 S C_x \tag{4.2}$$

Avec C_x le coefficient de traînée.

Ce dernier résulte de la friction et pression qui s'appliquent sur la fusée. Donc le C_x dépend de la forme de la fusée et du nombre de Mach. C'est pourquoi en subsonique il est considérée quasi constant et, de simples essais en soufflerie ont permis de le prouver et d'en déduire des expressions empiriques beaucoup travaillées. Le cas pour des vitesses supérieures est plus complexe. C'est un domaine moins travaillé par rapport au subsonique, et ainsi seuls les privilégiés accédant à des essais en soufflerie peuvent obtenir des résultats concrets. Toutefois, il existe des logiciels permettant d'appliquer des cas de ce genre et certains des scientifiques en tirent des expressions empiriques. Ces résultats, qui différent d'un auteur à un autre, sont loin d'être universels, mais il s'agit ici d'obtenir des résultats, qui pour le peu convainquant, serviront de modèle pour la forme de la fusée [8] et [13].

Le tableau 4.1 suivant résume la composition du coefficient de traînée :

TYPES	CAUSES
Traînée de frottement de surface	Viscosité cinématique de l'air
Traînée de pression	Géométrie du corps
Traînée de base et interférences	Échappements, sillages et protubérances
Traînée de vague	Supersonique (M>1)

TABLE 4.1 – Ensemble des traînées de la fusée

4.2.1 Traînée de Frottement

Traînée de Frottement du Corps

- *Le nombre de Reynold pour le corps de la fusée devient :*

$$R = \frac{aML}{12\nu}(1 + 0.0283M - 0.043M^2 + 0.2107M^3 - 0.03829M^4 + 0.002709M^5) \tag{4.3}$$

- *Le coefficient de frottement de la surface incompressible devient :*

$$Cf^{'} = 0.037036 \cdot R^{0.155079} \tag{4.4}$$

- *Le coefficient de frottement de la surface compressible devient :*

$$Cf = Cf^{'}(1 + 0.00798M - 0.1813M^2 + 0.0632M^3 - 0.00933M^4 + 0.000549M^5) \tag{4.5}$$

- *Le coefficient de frottement de la surface incompressible avec rugosité devient :*

$$Cf^{'}_{ro} = \frac{1}{[1.89 + 1.62log(\frac{L}{K})]^{2.5}} \tag{4.6}$$

K=0	smooth surface
K=0.00002 to 0.00008	polished metal or wood
K=0.00016	natural sheet metal
K=0.00025	smooth matte paint, carefully applied
K=0.0004 to 0.0012	standard camouflage paint

TABLE 4.2 – Valeurs de K

- *Le coefficient de frottement de la surface compressible avec rugosité devient :*

$$Cf_{ro} = \frac{Cf'_{ro}}{1 + 0.2044M^2} \tag{4.7}$$

- *Le coefficient de la surface finale devient :*

$$Cf_{final} = \begin{cases} Cf & : \text{si} \quad Cf \geq Cf_{ro} \\ Cf_{ro} & : \text{si} \quad Cf \leq Cf_{ro} \end{cases} \tag{4.8}$$

- *Le coefficient de traînée due à la friction devient :*

$$Cd_{f,body} = Cf_{final}[1 + \frac{60}{(\frac{L}{d})^3} + 0.0025(\frac{L}{d})]\frac{4S_B}{\pi d^2} \tag{4.9}$$

Traînée de Frottement des ailerons

- *Le nombre de Reynold pour les ailerons devient :*

$$R_1 = \frac{aMC_r}{12\nu}(1 + 0.0283M - 0.043M^2 + 0.2107M^3 - 0.03829M^4 + 0.002709M^5) \tag{4.10}$$

- *Le coefficient de frottement de la surface incompressible devient :*

$$Cf'_1 = 0.037036 \cdot R_1^{0.155079} \tag{4.11}$$

- *Le coefficient de frottement de la surface compressible devient :*

$$Cf_1 = Cf'_1(1 + 0.00798M - 0.1813M^2 + 0.0632M^3 - 0.00933M^4 + 0.000549M^5) \tag{4.12}$$

- *Le coefficient de frottement de la surface incompressible avec rugosité devient :*

$$Cf'_{ro,1} = \frac{1}{[1.89 + 1.62log(\frac{C_r}{K})]^{2.5}} \tag{4.13}$$

- *Le coefficient de frottement de la surface compressible avec rugosité devient :*

$$Cf_{ro,1} = \frac{Cf'_{ro,1}}{1 + 0.2044M^2} \tag{4.14}$$

- *Le coefficient de la surface finale devient :*

$$Cf_{final,1} = \begin{cases} Cf_1 & : \text{si} \quad Cf_1 \geq Cf_{ro,1} \\ Cf_{ro,1} & : \text{si} \quad Cf_1 \leq Cf_{ro,1} \end{cases} \tag{4.15}$$

- *Le nombre de Reynold incompressible pour les ailerons devient :*

$$R_{1,inc} = \frac{aMC_r}{12\nu} \tag{4.16}$$

$$\lambda = \frac{C_t}{C_r} \tag{4.17}$$

- *Le coefficient de frottement moyen pour chaque aileron devient :*

$$Cf_\lambda = Cf_{final,1} \cdot \frac{[log(R_1)]^{2.6}}{(\lambda^2 - 1)} \cdot (\frac{\lambda^2}{[log(R_1\lambda)]^{2.6}} - \frac{1}{[log(R_1)]^{2.6}} + 0.5646[\frac{\lambda^2}{[log(R_1\lambda)]^{3.6}} - \frac{1}{[log(R_1)]^{3.6}}]) \tag{4.18}$$

- *Le coefficient de traînée pour les ailerons devient :*

$$Cd_{f,fins} = Cf_\lambda \cdot [1 + 60(\frac{t}{C_r})^4 + 0.8(1 + 5\overline{X_{\frac{t}{c}}}^2)(\frac{t}{Cr})] \cdot \frac{4N_f S_f}{\pi d^2} \tag{4.19}$$

Avec $\overline{X_{\frac{t}{c}}} = \frac{X_{\frac{t}{c}}}{C_r}$ Avec $S_f \approx \frac{b}{2}(C_r + C_t)$

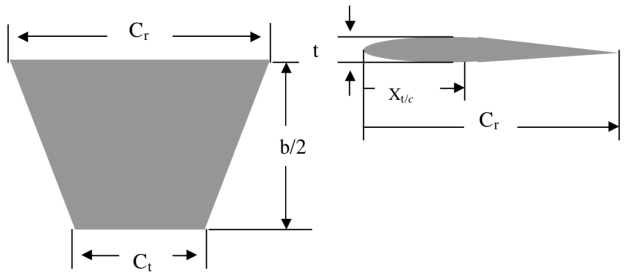


FIGURE 4.3 – Dimensions des fins de la fusée

Traînée de Friction de Protubérance

Une protubérance est une saillie arrondie faisant référence ici, au socle sur la partie extérieure de la fusée pour la maintenir sur sa base de décollage.

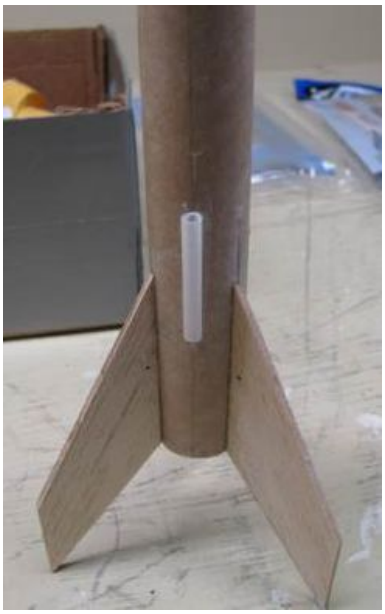


FIGURE 4.4 – Protubérance d'une fusée

- *Le nombre de Reynolds pour les protubérances devient :*

$$R_2 = \frac{aML_p}{12\nu}(1 + 0.0283M - 0.043M^2 + 0.2107M^3 - 0.03829M^4 + 0.002709M^5) \quad (4.20)$$

- *Le coefficient de frottement de la surface incompressible devient :*

$$Cf_2^{'} = 0.037036 \cdot R_2^{0.155079} \quad (4.21)$$

- *Le coefficient de frottement de la surface compressible devient :*

$$Cf_2 = Cf_2^{'}(1 + 0.00798M - 0.1813M^2 + 0.0632M^3 - 0.00933M^4 + 0.000549M^5) \quad (4.22)$$

- *Le coefficient de frottement de la surface incompressible avec rugosité devient :*

$$Cf_{ro,2}^{'} = \frac{1}{[1.89 + 1.62log(\frac{L}{K})]^{2.5}} \quad (4.23)$$

- *Le coefficient de frottement de la surface compressible avec rugosité devient :*

$$Cf_{ro,2} = \frac{Cf_{ro,2}^{'}}{1 + 0.2044M^2} \quad (4.24)$$

- *Le coefficient de la surface finale devient :*

$$Cf_{final,2} = \begin{cases} Cf_2 & : \text{si} \quad Cf_2 \geq Cf_{ro,2} \\ Cf_{ro,2} & : \text{si} \quad Cf_2 \leq Cf_{ro,2} \end{cases} \quad (4.25)$$

- *Le coefficient de frottement de protubérance devient :*

$$Cf_{pro} = 0.8151Cf_{final,2}(\frac{a}{L_p})^{-0.1243} \quad (4.26)$$

- *Le coefficient de trâinée pour les protubérances devient :*

$$Cd_{pro} = Cf_{pro} \cdot [1 + 1.798(\frac{\sqrt{A}}{L_p})^{\frac{3}{2}}] \cdot \frac{4S_p ro}{\pi d^2} \tag{4.27}$$

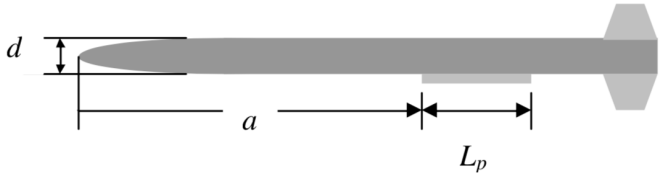


FIGURE 4.5 – Dimensions des launch lugs de la fusée

Coefficient de Frottement Total et de Traînée d'Interférence

La traînée d'interférence est provoquée par les turbulences qui proviennent de la rencontre d'écoulement aux directions différents. Elle est souvent provoquée au niveau des ailerons.

- *Le coefficient de trâinée due à la friction devient :*

$$Cd_F = Cd_{f,body} + K_F \cdot Cd_{f,fins} + K_F \cdot Cd_{pro} \tag{4.28}$$

Avec $K_F \approx 1.04$

4.2.2 Coefficient de Traînée de Base

Coefficient de Traînée de Base pour M < 0.6

- *Le coefficient de traînée de base pour M<0.6 devient :*

$$Cd_b(M < 0.6) = K_b \frac{(\frac{d_b}{d})^n}{\sqrt{Cd_F}} \tag{4.29}$$

avec $K_b = 0.0274 * \arctan((\frac{L_o}{d}) + 0.0116)$ et $n = 3.6542(\frac{L_o}{d})^{-0.2733}$

Coefficient de Traînée de Base pour M > 0.6

- *Le coefficient de traînée de base pour M>0.6 devient :*

$$Cd_b(M \geq 0.6) = Cd_b(M = 0.6) f_b \tag{4.30}$$

$f_b = 1.0 + 215.8(M - 0.6)^{6.0}$	For $0.6 < M < 1.0$
$f_b = 2.0881(M-1)^3 - 3.7938(M-1)^2 + 1.4618(M-1) + 1.883917$	For $1.0 < M < 2.0$
$f_b = 0.297(M-2)^3 - 0.7937(M-2)^2 - 0.1115(M-2) + 1.64006$	For $M > 2.0$

TABLE 4.3 – Valeurs de f_b

4.2.3 Coefficient de Traînée : Transonique

- *Le nombre de Mach de divergence en transsonique devient :*

$$M_D = -0.0156(\frac{L_N}{d})^2 + 0.136(\frac{L_N}{d}) + 0.6817 \tag{4.31}$$

- *Le nombre de Mach final dans la région transsonique devient :*

$$M_F = c_1(\frac{L_e}{d})^{c_2} + 1.0275 \tag{4.32}$$

$c_1 = 2.4$	For $(\frac{L_N}{L_e}) < 0.2$
$c_1 = -321.94(\frac{L_N}{L_e})^2 + 264.07(\frac{L_N}{L_e}) - 36.348$	For $(\frac{L_N}{L_e}) \geq 0.2$
$c_2 = -1.05$	For $(\frac{L_N}{L_e}) < 0.2$
$c_2 = 19.634(\frac{L_N}{L_e})^2 - 18.369(\frac{L_N}{L_e}) + 1.7434$	For $(\frac{L_N}{L_e}) \geq 0.2$

TABLE 4.4 – Valeurs de c_1 et c_2

- *L'augmentation maximale de la traînée sur la région transsonique devient :*

$$\Delta C_{Dmax} = \begin{cases} c_3(\frac{L_e}{d})^{c_4} & : For \quad (\frac{L_e}{d}) \geq 6 \\ c_3 * 6^{c_4} & : For \quad (\frac{L_e}{d}) < 6 \end{cases} \tag{4.33}$$

Avec $c_3 = 50.676(\frac{L_N}{L_e})^2 - 51.734(\frac{L_N}{L_b}) + 15.642$ et $c_4 = -2.2538(\frac{L_N}{L_e})^2 + 1.3108(\frac{L_N}{L_b}) - 1.7344$

- *L'élévation de traînée transsonique pour un nombre Mach devient :*

$$\Delta C_{dT} = \begin{cases} \Delta C_{Dmax} F & : For \quad M_D \leq M \leq M_F \\ 0 & : For \quad M_D > M \quad ou \quad M > M_F \end{cases} \tag{4.34}$$

Avec $F = -8.3474x^5 + 24.543x^4 - 24.946x^3 + 8.6321x^2 + 1.1195x$ et $x = [\frac{(M - M_D)}{M_F - M_D}]$

4.2.4 Coefficient de traînée : Supersonique

- *L'élévation de traînée supersonique pour un nombre Mach devient :*

$$\Delta C_{dS} = \begin{cases} \Delta C_{Dmax} & : For \quad M \geq M_F \\ 0 & : For \quad M < M_F \end{cases} \tag{4.35}$$

4.2.5 Coefficient de Traînée Totale

- *Le coefficient de trâinée totale devient :*

$$C_D = Cd_F + Cd_b + \Delta C_{dT} + \Delta C_{dS} \tag{4.36}$$

4.3 Résultat

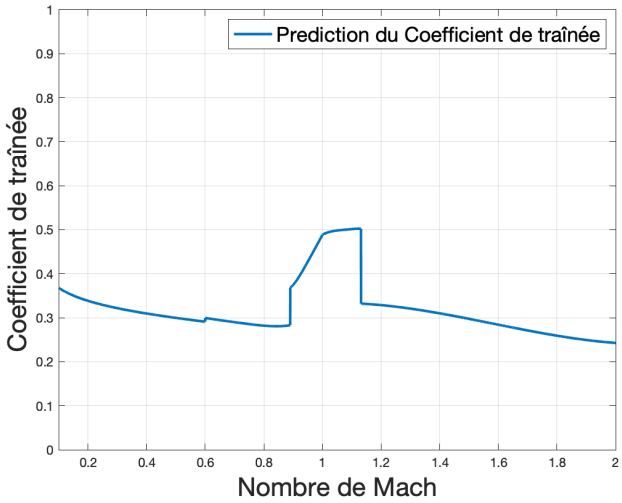


FIGURE 4.6 – Coefficient de Traînée en fonction du nombre de Mach

L'augmentation importante de la traînée est causée par la formation d'une onde de choc sur la surface du profil aérodynamique, ce qui peut entraîner une séparation de l'écoulement et des gradients de pressions défavorables. Cet effet exige que ce genre de profils destinés à voler à des vitesses supersoniques aient une poussée importante. Ainsi, cet effet est traduit par l'augmentation drastique du coefficient de traînée autour du nombre de Mach égal à 1. C'est à ce moment que le profil passe en transsonique. Lorsque la fusée passe en supersonique (M>1.2), l'onde de choc a traversé le profil entier pour se retrouver à ses deux extrémités : le système entier redevient à peu près stable.

Chapitre 5

Conclusion

Le modèle standardisé amélioré peut être considéré comme validé dans le sens où il est très proche des données récupérées par les radiosondes au Nouveau-Mexique (section 2.4) et en Suisse (section A.2).

L'interface graphique est fonctionnelle pour une simulation de vol en une dimension et elle peut encore être beaucoup améliorée au cours des années à venir, afin d'être la plus adaptée possible aux fusées de l'ERT.

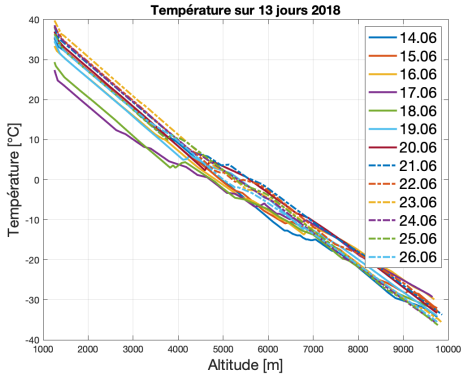
Concernant la prédiction des vols en supersonique de la fusée, de nombreuses études et tests doivent encore être faits avant de pouvoir affirmer que nous pouvons prédire avec précision sa trajectoire et toutes les caractéristiques de son vol. En effet, bien que les résultats obtenus avec les fonctions empiriques de la section 4.2 soit cohérents (cf. Figure 4.6), il n'est pas possible de valider ce modèle par manque d'essais. Un test en soufflerie est nécessaire, suivi d'une validation par logiciel. Ainsi, le supersonique reste un domaine complexe et peu recherché, avec dans notre cas un manque d'accès à des ressources et des logiciels comme Fluent d'Ansys afin de pouvoir progresser.

Annexe A

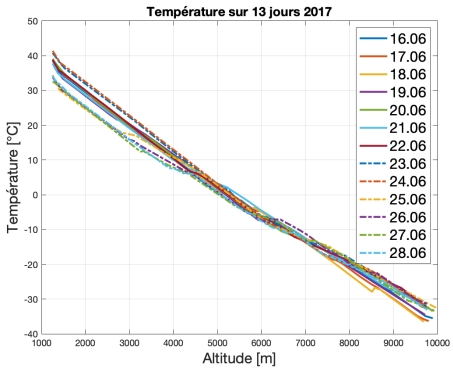
Atmosphère Standard

A.1 Données Radiosonde de température

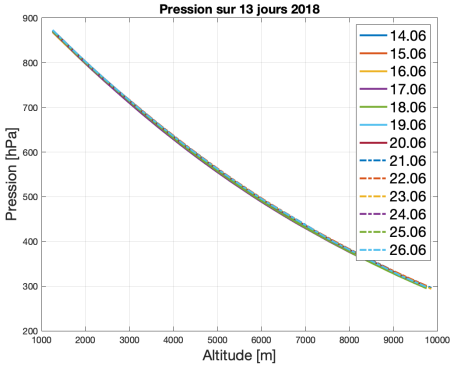
La figure (2.2) nous donne une variation de la température en fonction de l'altitude mais seulement le 20 juin de chaque année (2009->2018). Pour être certain de cette variation, nous avons comparer la variation de la température sur une plage de jour centrée sur le jour de la compétition(20 juin 2018 et 22 juin 2017) :



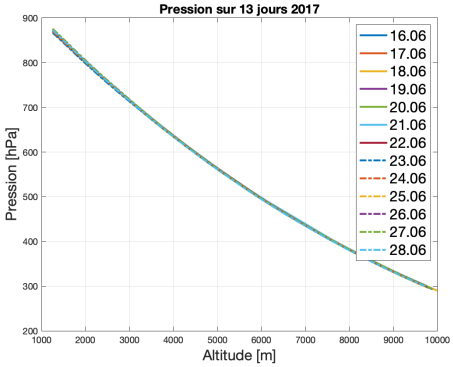
(a) Variation de la température sur une plage de 13 jours centrés sur le jour du lancement 2018



(b) Variation de la température sur une plage de 13 jours centrés sur le jour du lancement 2017



(c) Variation de la pression sur une plage de 13 jours centrés sur le jour du lancement 2018



(d) Variation de la pression sur une plage de 13 jours centrés sur le jour du lancement 2017

FIGURE A.1 – Données de radiosonde à Santa-Teresa [5]

On remarque qu'une fois de plus la température varie linéairement, avec une pente de

dT/dh =-9.5 [°C/km].

A.2 Atmosphère standard à Payerne

De nombreux tests de lancement se font à Cernier en Suisse [14]. Ce sont des lancés qui ne dépassent pas 2000 mètres d'altitude. Ainsi, nous décidons de vérifier l'allure du nouveau modèle atmosphérique sur ce lieu. Malheureusement, la Suisse ne possède qu'un centre d'envoi de radiosonde et il se situe à Payerne. Nous utiliserons donc ces références.

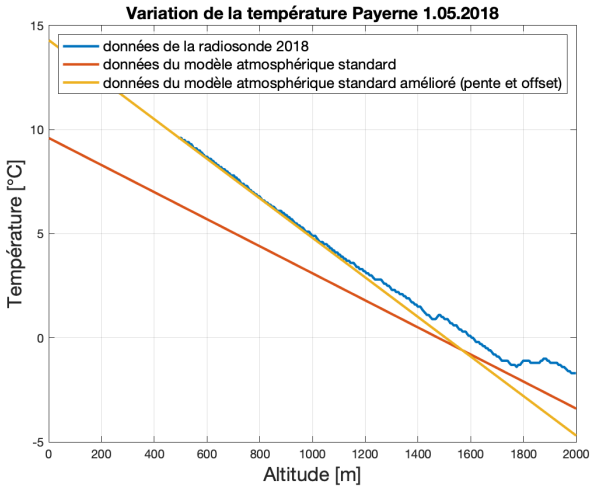


FIGURE A.2 – Variation de la température en fonction de l'altitude à Payerne

La simulation était relativement fausse pour les précédents lancés. Le nouveau modèle atmosphérique colle bien avec les données enregistrées en radiosonde. On fait de même avec la pression et la densité.

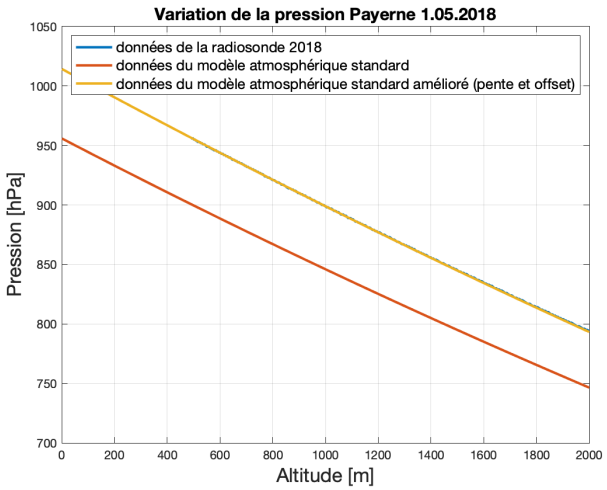


FIGURE A.3 – Variation de la pression en fonction de l'altitude à Payerne

Les données radiosondes et le modèle standard amélioré sont superposés sur la figure A.3

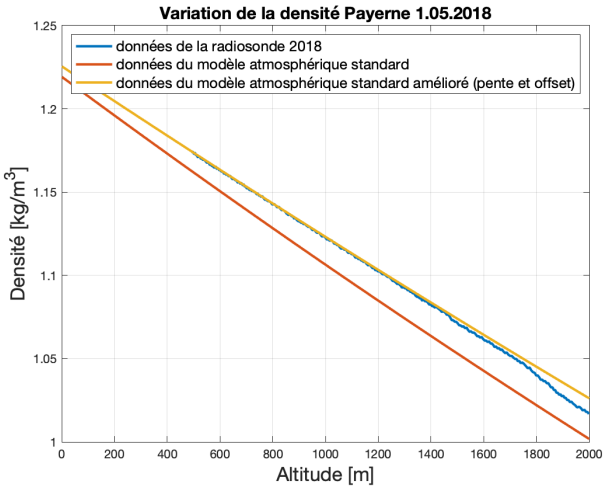


FIGURE A.4 – Variation de la densité en fonction de l'altitude à Payerne

Annexe B

Supersonique

B.1 Choix du moteur

Dans le cas du supersonique, nous devons choisir un moteur [12] permettant à la fusée d'atteindre l'objectif du concours : 30000 ft.

Nom moteur	Impulsion	Vitesse max [m/s]	Alt apogée [m]	Taille [mm]
O 25,000 VM P	30907	470	6932	1239
N 5800 CS P	20368	522	6852	1239
N 5600WT-P	13633	390	6683	1010
N 4800 T-P	19274	475	6415	1201
N 4800 T-0	19274	475	6415	1194
N4100-RL-0	17735	447	5750	1239

Table B.1 – Choix du moteur

Étant donné que la fusée actuelle a un diamètre de 98 mm, notre choix de moteur reste limité, surtout pour atteindre une altitude de 9144 m pour le concours. Les dimensions d'une fusée ayant pour objectif 9144 m seront ré-adaptées aux besoins du moteur.

B.2 Comparaison des Cd_x

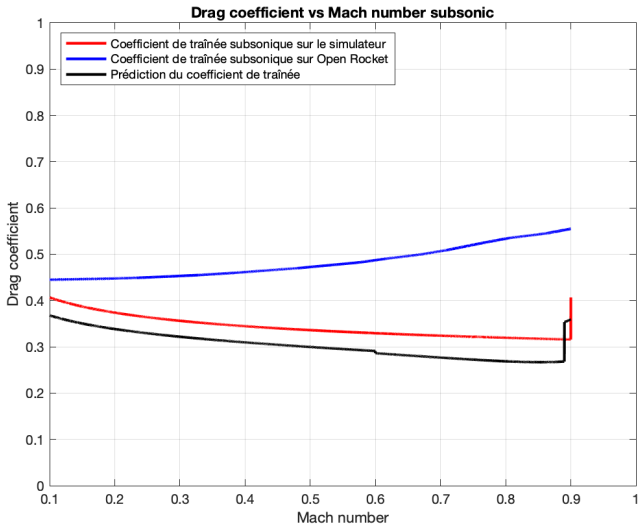


FIGURE B.1 – Comportement du coefficient de traînée en subsonique pour différents cas

On observe les courbes rouge (qui découle des travaux d'Emilien et Eric [6]) et noire (qui découle des équations de la section 4.2) ont la même allure et décroissent légèrement. Ce phénomène peut s'expliquer par le fait qu'à mesure que la vitesse augmente, il a été montré que le nombre Reynolds augmente énormément, signe d'une viscosité qui diminue. Or on a vu qu'avec la table 4.1, le coefficient de traînée en dépendait. Ainsi, ce terme deviendrait négligeable avec l'augmentation du nombre de Mach. De plus, on peut le justifier en prenant les gaz parfaits, entraînant un écoulement isentropique (c'est-à-dire viscosité négligée) pour des études supersoniques. Enfin la courbe bleue vient du simulateur Opensource *OpenRocket* [9] qui n'est pas si fiable si l'on en croit les résultats des deux courbes précédentes. D'où l'importance de créer notre propre simulateur pour être le plus optimal possible.

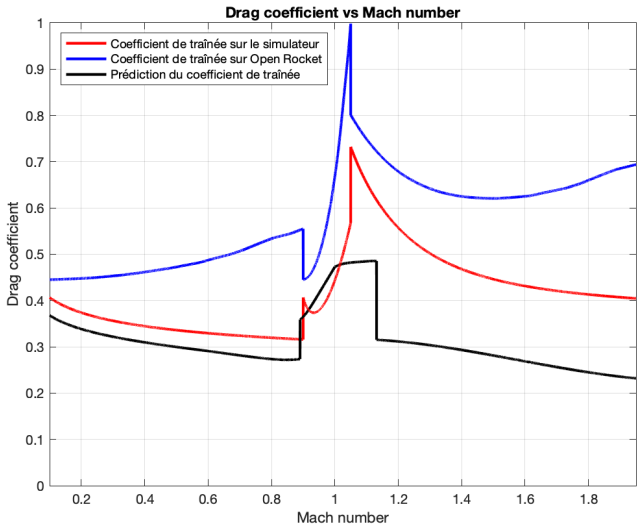
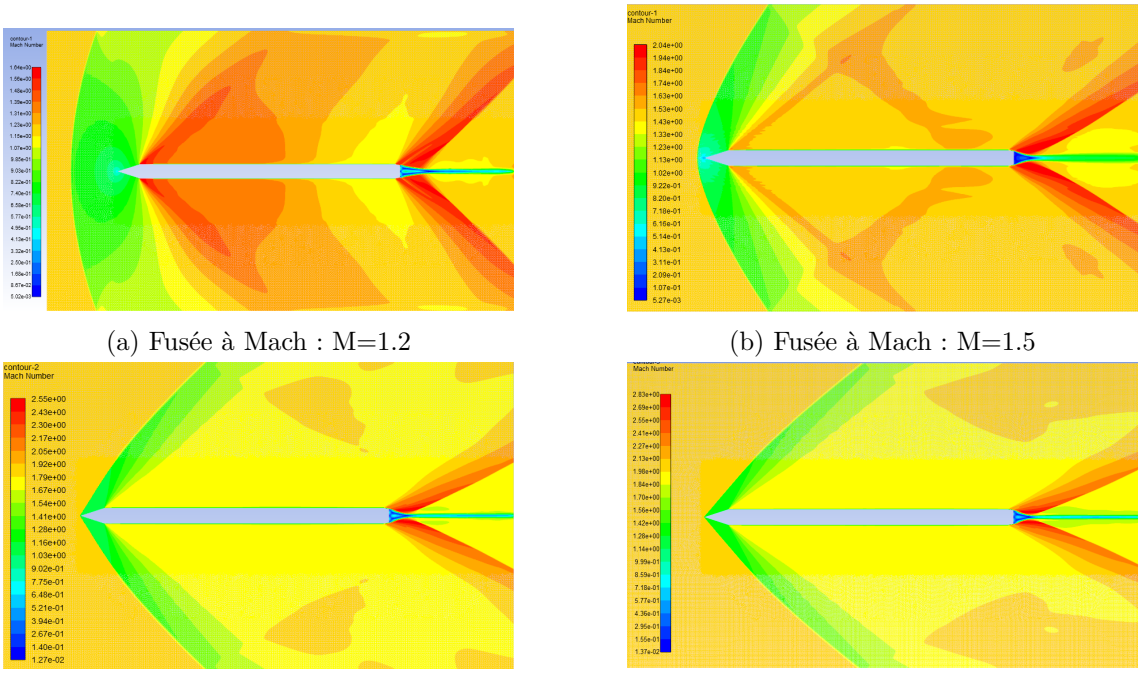


FIGURE B.2 – Comportement du coefficient de traînée pour différents cas

B.3 Résultats de Fluent Ansys

Afin de vérifier la théorie exploitée dans les sections précédentes, il est possible d'utiliser un logiciel qui exploite les éléments finis pour une précision absolue. Les figures suivantes proviennent d'une simulation 2D pour établir le comportement de l'air à des vitesses supersoniques. On remarque à mesure que la fusée augmente sa vitesse, une onde de choc vient se déposer sur chaque extrémité.



(a) Fusée à Mach : M=1.2

(b) Fusée à Mach : M=1.5

(c) Fusée à Mach : M=1.8

(d) Fusée à Mach : M=2.0

FIGURE B.3 – Comportement de l'air à des vitesses supersoniques

Annexe C

Analyses de stabilité

Il y a plusieurs critères à évaluer avant de permettre à une fusée de voler. Deux d'entre eux sont la marge de stabilité au cours du vol et le coefficient d'amortissement de la fusée. Ceux-ci permettent en effet d'affirmer si la fusée aura une trajectoire de vol nominale.

La marge de stabilité est définie comme la distance entre le centre de masse et le centre de pression de la fusée à un instant donné, normalisé par le diamètre de référence de la fusée, i.e. son diamètre maximal.

Le coefficient d'amortissement de la fusée est défini par Mandell [10] par l'équation (C.2) et détaillé dans [2], [3] et [4], en émettant l'hypothèse que le moment d'inertie longitudinal est constant et que la fusée n'a pas de rotation le long de son axe de vol (pas de roll). L'équation résolue est ainsi (C.1).

$$I_L \frac{d^2\alpha}{dt^2} + C_2 \frac{d\alpha}{dt} + C_1 \alpha = 0 \tag{C.1}$$

$$\zeta = \frac{C_2}{2\sqrt{C_1 I_L}} \tag{C.2}$$

I_L est le moment d'inertie longitudinal de la fusée,

C_1 est le coefficient de moment correctif, défini par (C.3) et expliqué par [2],

C_2 est le coefficient de moment d'amortissement, défini par (C.4) et expliqué par [3] et [4].

$$C_1 = \rho/2 \times V^2 \times A_r \times C_{N\alpha} \times (Z - W) \tag{C.3}$$

ρ est la densité de l'air en $[\mathrm{kg\,m^{-3}}]$,

V est la vitesse de la fusée en $[\mathrm{m\,s^{-1}}]$,

A_r est l'aire de référence de la fusée, calculée à partir du diamètre maximal de celle-ci par $A_r = \pi \times d_{max}^2/4$,

$C_{N\alpha}$ est le coefficient de force normale de la fusée, dont les équations afin de le calculer en fonction de la géométrie de la fusée sont fournies ici [1],

Z est la distance du centre de pression au bout du nez de la fusée,

W est la distance du centre de masse au bout du nez de la fusée.

$$C_2 = C_{2A} + C_{2R} \tag{C.4}$$

$$= \frac{\rho \times V \times A_r}{2} \sum \left(C_{N\alpha_{composant}} \times [Z_{composant} - W]^2 \right) + \dot{m} \times [L_{ne} - W]^2 \tag{C.5}$$

C_{2A} est le coefficient du moment d'amortissement aérodynamique

C_{2R} est le coefficient du moment d'amortissement propulsif

$C_{N\alpha_{composant}}$ est le coefficient de force normale du composant de la fusée, dont les équations sont fournies ici [1],

$Z_{composant}$ est la distance du centre de pression du composant au bout du nez de la fusée,

$\dot{m}$ est le flux d'expulsion de masse par la tuyère, en $[\mathrm{g\,s^{-1}}]$

L_{ne} est la distance entre le plan d'éjection des gaz de la tuyère et le nez de la fusée

Table des figures

Liste des tableaux

References

[1] James S. Barrowman et Judith A. Barrowman. *The Theoretical Prediction of the Center of Pressure*. English. NARAM-8 R&D Project. Août 1996, p. 1–43. URL : https://www.apogeerockets.com/downloads/barrowman_report.pdf.

[2] "Basics of Dynamic Flight Analysis (Part 2) - The Corrective Moment Coefficient". English. In : *Apogee Peak of Flight* 193 (sept. 2007), p. 2–7. URL : https://www.apogeerockets.com/education/downloads/Newsletter193.pdf.

[3] "Basics of Dynamic Flight Analysis (Part 3) - The Damping Moment Coefficient". English. In : *Apogee Peak of Flight* 195 (oct. 2007), p. 2–7. URL : https://www.apogeerockets.com/education/downloads/Newsletter195.pdf.

[4] "Basics of Dynamic Flight Analysis (Part 3) - The Damping Ratio". English. In : *Apogee Peak of Flight* 197 (nov. 2007), p. 2–7. URL : https://www.apogeerockets.com/education/downloads/Newsletter197.pdf.

[5] *Departement of Atmospheric Science : University of Wyoming, College of Engineering*. en. DOI : http://weather.uwyo.edu/upperair/sounding.html.

[6] Eric Brunner et Emilien Mingard. "Simulation avancée de la trajectoire d'une fusée et application du contrôle actif". In : (2017).

[7] Jr. H.H. Hurt. "Aerodynamics for Naval Aviators". In : (1965).

[8] Sighard F Hoerner. "Fluid Dynamic Drag,theoretical, experimental and statistical information". In : (1965), p. 282–445.

[9] *Logiciel OPENROCKET*. en. DOI : http://openrocket.info.

[10] Gordon K Mandell, George J Caporaso et William P Bengen. *Topics in advanced model rocketry*. English. OCLC : 67714804. Cambridge, Mass. : MIT Press, 1990.

[11] Planète Sciences. "Le Vol de la Fusée, Stabilité et Trajectographie". In : (juillet 2018).

[12] *Thrustcurve, Motor Performance Data Online*. en. DOI : http://www.thrustcurve.org.

[13] Unknown. "Drag Coefficient Prediction,Chapter 1". In : ().

[14] *UQAM Weather Center, meteocentre*. en. DOI : http://meteocentre.com/radiosonde/get_sounding.php?stn=72364&type=rs&yyyy=2019&mm=02&dd=20&run=00&hist=0&show=0&lang=en&area=u.