ECOLE POLYTECHNIQUE DE BRUXELLES

202526

Projet Tran-H101

L'Eaudyssée

Sacha Zarella

Maxime Mfaddel

James Obresse

Daniel Borosean

Thibault Musschebroek

Florian Martin

Patrick Simon

2025

Table des matières

1 Résumé exécutif

Depuis plusieurs décennies la recherche spatiale est au cœur des intérêts scientifiques. La conception d'une fusée à eau permet d'avoir une compréhension simplifiée du fonctionnement d'une véritable fusée, en permettant notamment l'étude et la modélisation des principes fondamentaux de la propulsion, de l'aérodynamisme et du mouvement, avec des ressources limitées.

Afin de modéliser le déplacement d'une telle fusée, un diagramme du corps libre de la fusée volant dans l'air est nécessaire 2.2. Par la suite, nous avons opté pour la deuxième loi de Newton pour un système à masse variable(plus précisément pour le théorème de la quantité de mouvement 2.2) dans le but de déterminer l'accélération de la fusée. Pour utiliser cette dernière, il convient de déterminer les forces extérieures s'exerçant sur la fusée, ainsi que la force de poussée de la fusée. Les forces extérieures sont déterminées par l'équation de la traînée aérodynamique 2.11 et par l'équation de la force gravitationnelle 2.12. La force de poussée quant à elle, dépendant de la vitesse d'éjection de l'eau est déterminée en utilisant l'équation de Bernoulli 1.3. Certaines hypothèses doivent être posées afin de permettre l'utilisation des équations de base 2.1.2. De plus, pour déterminer certaines données, telles que le coefficient de traînée de la fusée 3.4, une approche empirique est utilisée. Après des tests réalisés sur le modèle de prédiction, certains paramètres ayant un effet négligeable2.1.3 sur le modèle ont été retirés (notamment dans l'équation de Bernoulli).

Des matériaux légers ont été utilisés dans la conception de la fusée afin de minimiser sa masse. Le corps de la fusée 4.1 est composé de deux bouteilles PET (réservoir et coiffe) 3.3 imbriquées l'une dans l'autre ainsi que d'un set de quatre ailerons découpés dans une plaque de plastique. Les différentes pièces sont attachées par de la colle forte (froide), de l'adhésif ou de la ficelle. La colle chaude n'est pas utilisée afin de ne pas endommager le réservoir de la fusée.

La coiffe 3.3.2 sert à contenir la charge utile et le parachute. Cette dernière est elle-même divisée en 2 sous-compartiments séparés entre eux ayant chacun le rôle respectif évoqué précédemment. Le premier sous-compartiment est accessible par l'extérieur de la fusée à l'aide d'une trappe et le deuxième s'ouvre entièrement afin de déployer le parachute. Ces deux systèmes d'ouverture

sont conçus sans utiliser de pièces lourdes ou volumineuses et permettent de conserver une masse totale basse et un profil aérodynamique avantageux.

Le parachute 3.2 de la fusée se compose d'une bâche en plastique attachée à quatre points de la fusée et renforcée sur les coins. Des tests de chute réalisés avec une masse ajoutée de 100g (simulant la présence de l'altimètre et de la charge utile) ont montré l'efficacité du parachute. Afin de rendre le système de déploiement du parachute simple et léger, le sous-compartiment contenant le parachute est attaché au reste de la fusée de manière à s'ouvrir et permettre le déploiement du parachute lorsque la fusée arrive dans sa phase de redescente.

La forme et le placement optimal des ailerons 3.1 sont déterminés à l'aide de recherches théoriques. Ces derniers sont découpés dans une plaque de plastique et sont donc facilement redécouplables (ajustables). Cette fonction d'ajustement des ailerons permet de corriger d'éventuels problèmes émergents liés à une surstabilité de la fusée durant le vol.

Pour le moment, deux expériences ont été réalisées avec notre prototype. La première, le test de poussée 3.5, nous permet de valider notre modèle théorique, car les résultats obtenus lors de la résolution numérique du modèle correspondent bien avec les résultats obtenus lors du test de poussée. La deuxième, la colonne d'eau 3.4, nous permet de déterminer empiriquement le coefficient de frottement de notre fusée.

Il est clair qu'il reste encore du travail pour l'évaluation finale. Il faut travailler sur la partie écoconception, budget et rendre facile et rapide le démontage/ remontage. De plus, vu que le prototype aura surement évolué d'ici là, il faudra refaire les tests de colonne d'eau et de poussée pour qu'ils soient à jour.

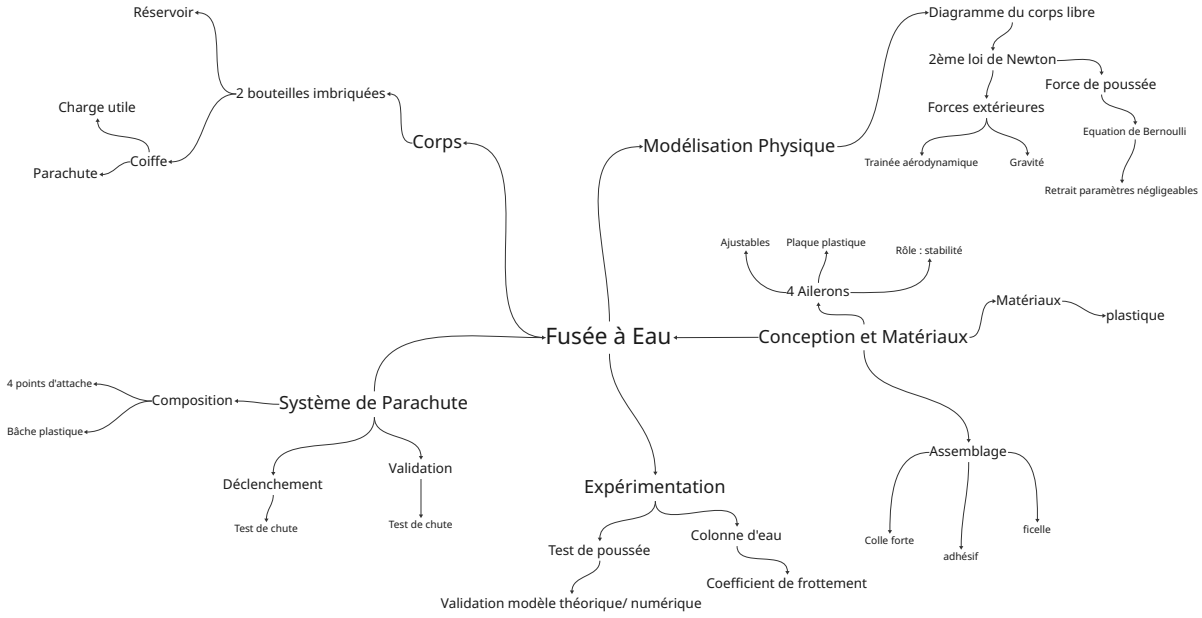

FIGURE 1.1: Le mindmap de notre projet

2 Modèle

Symbole	Description
P_{atm}	Pression atmosphérique
$P_{init-relative}$	Pression initiale relative dans la bouteille
P_{init}	Pression initiale absolue dans la bouteille
ρ_{air}	Masse volumique de l'air
ρ_{eau}	Masse volumique de l'eau
g	Accélération gravitationnelle
γ	Coefficient adiabatique de l'air
V_b	Volume total de la bouteille
A_b	Aire de section interne de la bouteille
A_{eau}	Aire de l'orifice de sortie de l'eau
$m_{\text{fusée}}$	Masse de la fusée vide
C_d	Coefficient de traînée aérodynamique
$V_{\text{eau,init}}$	Volume initial d'eau
$V_{\text{air,init}}$	Volume initial d'air
$\rho_{\text{air,init}}$	Masse volumique de l'air initial
$m_s$air,init	Masse d'air initiale dans la bouteille
z	Altitude de la fusée (variable d'état)
v	Vitesse verticale de la fusée (variable d'état)
$V_{\text{eau}}(t)$	Volume d'eau restant (variable d'état)
V_{air}	Volume d'air instantané
$P(t)$	Pression absolue à l'instant t

Symbole	Description
$V(t)$	Volume de l'air à l'instant t
$h(t)$	Hauteur de la colonne d'eau
P_{sortie}	Pression manométrique de sortie
v_{eau}	Vitesse d'éjection
v_w	Vitesse de l'eau dans la fusée
F_{poussee}	Force de poussée
F_g	Force de gravité
F_{fr}	Force de traînée
m_{total}	Masse totale instantanée
dV_{eau}/dt	Débit volumique
$\frac{dv}{dt}$	Accélération
$\frac{dz}{dt}$	Vitesse verticale
$\frac{dm}{dt}$	Débit massique
p	Quantité de mouvement
F_{ext}	Force extérieure

TABLE 2.1: Liste des variables

2.1 Force de poussée

2.1.1 Objectif

Ce modèle nous permet de déterminer la force de poussée qui sera générée lors de la phase d'éjection de l'eau. Il dépend des dimensions de la bouteille (réservoir), de la pression interne introduite et du volume d'eau. Il permet de choisir et d'optimier la pression et le volume d'eau initiaux au sein du réservoir afin d'obtenir les résultats souhaités lors du décollage.

2.1.2 Hypothèses

Pour pouvoir développer ce modèle il est nécessaire de faire certaines hypothèses :

— L'eau est un liquide incompressible.

— L'air au sein de la bouteille est un gaz parfait.

— Le modèle se déroule dans des conditions adiabatiques, il n'y a pas d'échange de chaleur.

— L'eau est un fluide parfait, sa viscosité est nulle.

2.1.3 Formulation mathématique

Pour débuter le modèle théorique nous nous sommes basés sur la formule de poussée ([Whe02]) :

$$F_{poussee} = v_{eau}\frac{dm}{dt} \tag{1.1}$$

Le débit massique étant donné par l'équation (source) :

$$\frac{dm}{dt} = A_e \cdot \rho_{eau} \cdot v_{eau} \tag{1.2}$$

Il nous faut encore déterminer la vitesse d'éjection de l'eau en utilisant bernoulli (source) :

$$P_1 + \frac{1}{2}\rho_{eau} \cdot v_1^2 + \rho_{eau} \cdot g \cdot h_1 = P_2 + \frac{1}{2}\rho_{eau} \cdot v_2^2 + \rho_{eau} \cdot g \cdot h_2 \tag{1.3}$$

La vitesse au point 2 est considérée comme nulle car elle est infiniment petite par rapport à celle du point 2.

$$\frac{dV_1}{dt} = \frac{dV_2}{dt} \tag{1.4}$$

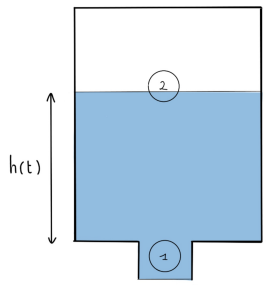

FIGURE 2.1: Bernoulli dans la bouteille

$$\frac{A_b \cdot h(t)}{dt} = \frac{A_e \cdot h(t)}{dt} \tag{1.5}$$

$$A_b \cdot v_1 = A_e \cdot v_2 \tag{1.6}$$

$$v_2 = \frac{A_e}{A_b} \cdot v_1 \tag{1.7}$$

$$v_w = \frac{A_e}{A_b} \cdot v_{eau} \tag{1.8}$$

Le rapport de $\frac{A_e}{A_b}$ est de l'ordre du pourcent et peut donc etre négligé.

$$P_{atm} + \frac{1}{2}\rho_{eau} \cdot v_{eau}^2 = P(t) + \rho_{eau} \cdot g \cdot h(t) \tag{1.9}$$

$$v_{eau} = \sqrt{\frac{2(P(t) - P_{atm} + \rho_{eau} \cdot g \cdot h(t))}{\rho_{eau}}} \tag{1.10}$$

Il ne reste plus qu'à déterminer P(t) via la loi de Laplace alternative (source)

$$P_1 V_1^\gamma = P_2 V_2^\gamma \tag{1.11}$$

$$P(t)V(t)^\gamma = P_{init} V_{air,init}^\gamma \tag{1.12}$$

$$P(t) = P_{init} \cdot (\frac{V_{air,init}}{V(t)})^\gamma \tag{1.13}$$

$$P(t) = P_{init} \cdot (\frac{V_b - V_{eau,init}}{V_b - A_b \cdot h(t)})^\gamma \tag{1.14}$$

γ est déterminé par :

$$\gamma = 1 + \frac{2}{n_d} \tag{1.15}$$

En ayant considéré que l'eau est un fluide incompressible, n_d vaut 5. Et donc $\gamma = 1,4$

Ce qui permet d'obtenir :

$$F_{poussee} = 2 \cdot A_e \cdot (P_{init} \cdot (\frac{V_b - V_{eau,init}}{V_b - A_b \cdot h(t)})^{\gamma} - p_{atm} + \rho_{eau} \cdot g \cdot h(t)) \tag{1.16}$$

Ce modèle est idéalisé. En réalité, l'eau n'est pas éjéctée parfaitement verticalement du réservoir.

2.2 Equation du mouvement

2.2.1 Objectif

L'équation du mouvement nous permet de déterminer la hauteur et la vitesse de la fusée à tout instant. Elle dépend de la forme et des dimensions de la fusée, de la pression et du volume initial au sein du réservoir.

2.2.2 Hypothèses

Nous faisons l'hypothèse que le mouvement est dirigé parfaitement verticalement.

2.2.3 Formulation mathématique

Pour débuter le modèle théorique nous sommes partis du théorème de la quantité de mouvement (source) :

$$\frac{dp}{dt} = \sum F_{ext} \tag{2.1}$$

Les forces extérieures sont identifiables sur le diagramme du corps libre de la fusée.

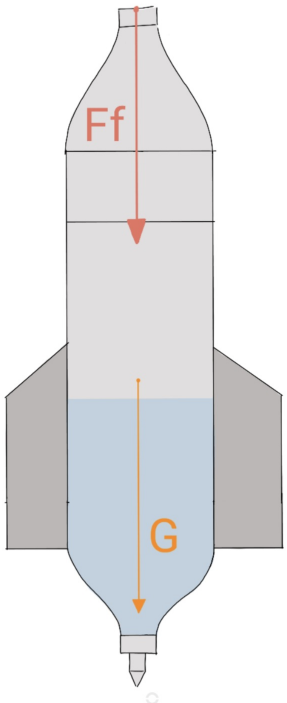

FIGURE 2.2: Diagramme du Corps Libre

À partir de la définition de la quantité de mouvement il est possible d'exprimer l'accélération de la fusée en fonction de l'ensemble des forces s'exerçant sur cette dernière.

$$p(t) = m(t) \cdot v(t) \tag{2.2}$$

$$p(t+dt) = m(t+dt) \cdot v(t+dt) + dm \cdot (v - ve_{eau}) \tag{2.3}$$

$$p(t+dt) = (m_{total} - dm) \cdot (v + dv) + dm \cdot (v - v_{eau}) \tag{2.4}$$

$$p(t+dt) = m_{total} \cdot v + dv \cdot m_{total} - dm \cdot v - dm \cdot dv + dm \cdot v - dm \cdot v_{eau} \tag{2.5}$$

Les infinitésimaux d'ordre supérieur sont négligeables ($dm \cdot dv$).

$$p(t+dt) = m_{total} \cdot v + dv \cdot m_{total} - dm \cdot v_{eau} \tag{2.6}$$

$$dp = p(t+dt) - p(t) \tag{2.7}$$

En remplacant $p(t)$ et $p(t+dt)$ dans l'équations ci dessus nous obtenons :

$$dp = dv \cdot m_{total} - dm \cdot v_{eau} \tag{2.8}$$

On peut finalement écrire :

$$\frac{dv}{dt} = \frac{\sum F_{ext} + \frac{dm}{dt} \cdot v_{eau}}{m_{total}} \tag{2.9}$$

Avec $\sum F_{ext}$ valant :

$$\sum F_{ext} = -F_g - F_{fr} \tag{2.10}$$

Il nous reste plus qu'à insérer F_{fr} (source) et F_g pour finaliser notre modèle.

$$F_{fr} = \frac{1}{2} \cdot \rho_{air} \cdot C_d \cdot A_b \cdot v^2 \tag{2.11}$$

$$F_g = m_{total} \cdot g \tag{2.12}$$

Pour conclure, l'équation compète s'exprime de la manière suivante

$$\frac{dv}{dt} = \frac{\frac{1}{2} \cdot \rho_{air} \cdot C_d \cdot A_b \cdot v^2 + m_{total} \cdot g + 2 \cdot A_e \cdot (P_{init} \cdot (\frac{V_b - V_{eau,init}}{V_b - A_b \cdot h(t)})^\gamma - p_{atm} + \rho_{eau} \cdot g \cdot h(t))}{m_{total}} \quad (2.12)$$

En intégrant une première fois nous obtenons la vitesse. En intégrant une deuxième fois nous obtenons l'accélération.

3 Présentation des résultats clés

3.1 Ailerons

3.1.1 Objectif général

Les ailerons ont de multiples utilités dans une fusée. Nous avons retenu les deux plus importantes.
Premièrement, de reculer le centre de pression en amenant des frottements à l'arrière de la fusée. Deuxièmement, stabiliser la fusée car les ailerons poussent dans le sens opposé à l'angle d'attaque. Des ailerons optimaux maximisent leur surface (responsable de la stabilisation) tout en minimisant leur poids et leur frottement aérodynamique. Le nombre d'ailerons, tant qu'il est supérieur à trois, n'a que peu d'impact. [Nak24]

3.1.2 Premier prototype

Objectif : corriger la trajectoire de la fusée en garantissant une répartition symétrique des surfaces de contrôle.

Prototype testé : le premier prototype comportait quatre ailerons fixés à un anneau circulaire. Nous avons opté pour ce design afin d'obtenir une géométrie parfaitement régulière, permettant ainsi d'avoir une bonne stabilité aérodynamique.

Résultat du test : le diamètre de l'anneau était trop petit et ne pouvait donc pas être fixé à la fusée.

Conclusion et évolution : l'anneau ne savait pas glisser sur la bouteille. Nous avons décidé d'agrandir le diamètre l'anneau qui reliait les ailerons.

3.1.3 Deuxième prototype

Objectif : agrandir l'anneau pour laisser passer la bouteille.

Prototype testé : ce deuxième prototype devait nous permettre d'adapter nos ailerons au diamètre de la fusée. La solution que nous avons trouvée était de couper l'anneau en 4. Chaque partie étant composée d'un quart d'anneau et d'un aileron. Ce montage devait nous servir à attacher les ailerons à la bouteille et à garder l'anneau de support, qui permettait de stabiliser les ailerons et d'augmenter la surface de contact avec la bouteille.

Résultat du test : un des ailerons a cassé pendant que l'anneau se faisait découper.

Conclusion et évolution : pendant nos recherches théoriques, nous étions arrivés à la conclusion que le choix du nombre d'ailerons n'avait pas beaucoup d'importance et que 3 comme 4 ailerons pouvaient remplir le rôle. L'avantage de 3 ailerons comparé à une solution à 4 était un poids plus léger et moins de frottements aérodynamiques.

3.1.4 Troisième prototype

Objectif : passer sous une forme à 3 ailerons.

Prototype testé : le troisième prototype comportait 3 ailerons, séparés par des angles de 120°.

Résultat du test : les ailerons étaient attachés via du scotch à la bouteille.

Conclusion et évolution : ces ailerons remplissaient leur rôle mais ils n'apportaient pas les avantages qu'une version à 3 ailerons contre 4 devaient apporter. La méthode pour les fixer (du scotch) était peu aérodynamique (surface rugueuse et avec air emprisonné entre les épaisseurs de scotch). De plus, le poids rajouté ne permettait plus de justifier la solution à 3 ailerons.

3.1.5 Quatrième prototype

Objectif : revenir à notre solution originale.

Prototype testé : le quatrième prototype était similaire en tout point au premier mais avait un anneau plus large.

Résultat du test : les ailerons remplissaient parfaitement leur rôle.

Conclusion et évolution : nous sommes restés sur cette version.

3.2 Parachute

3.2.1 Objectif général

L'objectif général est d'assurer un déploiement fiable du parachute tout en garantissant une vitesse de descente suffisamment faible pour la fusée. Le critère le plus difficile à atteindre est un déploiement fiable et constant. Nos recherches théoriques nous ont fait comprendre qu'un trou au milieu du parachute d'une surface d'à peu près un dixième de la surface totale du parachute avait tendance à fortement améliorer la résistance aux vents latéraux et la stabilité globale de celui-ci.

3.2.2 Premier prototype : parachute circulaire conique

Prototype testé : le premier parachute était de forme circulaire avec un orifice central. Un secteur avait été retiré du cercle afin de lui donner une forme légèrement conique une fois les rayons découpés et recollés entre eux.

Résultats des tests : plusieurs essais de déploiement ont été réalisés avec différents pliages. Le parachute ne s'est jamais correctement déplié. la forme conique compliquait fortement tout pliage.

Conclusion : la géométrie conique n'est pas adaptée car elle diminue considérablement la fiabilité du déploiement. Cette forme a donc été abandonnée.

3.2.3 Second prototype : parachute carré

Prototype testé : un parachute carré avec un trou central carré a été conçu afin de simplifier le pliage et d'améliorer la probabilité d'ouverture.

Résultats des tests : ce parachute se déployait correctement lors des essais, remplissant ainsi la fonction attendue.

Limite identifiée : après l'ajout d'autres composants (notamment les ailerons), la fusée est devenue plus lourde et la surface du parachute s'est révélée insuffisante pour générer le ralentissement nécessaire.

Conclusion : une augmentation de la surface était nécessaire.

3.2.4 Troisième prototype : parachute carré agrandi

Prototype testé : la géométrie carrée avec trou central a été conservée, mais la surface totale du parachute a été augmentée.

Résultat clé : cette version génère une traînée suffisante pour assurer une descente sécurisée, même avec la masse accrue de la fusée. Le problème est qu'il ne se déploie pas de manière fiable. Nous voyons cependant que même à moitié déployé, il arrive à ralentir le tout suffisamment pour éviter un crash. Nous essayerons dans nos prochaines améliorations, de changer d'autres paramètres tels que la longueur des fils et le ratio entre surface/trou. La surface totale peut aussi probablement être optimisée.

3.3 Compartiments

3.3.1 masse utile

Prototype : le compartiment de la masse utile est constitué de la partie inférieure d'une bouteille en plastique découpée. Il possède une ouverture carée qui permet d'y placer la masse utile par l'extérieur de la fusée. Il peut être fermé grâce à un système léger de trombones fixés sur la paroi, permettant une fermeture sécurisée tout en concèdant un remplacement rapide de la masse.

Résultat : le système s'est montré suffisamment rigide et résistant aux vibrations du vol tout en conservant une bonne accessibilité pour l'installation de la charge utile.

Conclusion et améliorations : conservation de ce système.

3.3.2 Coiffe

Prototype : la coiffe est constituée de la partie supérieure de la même bouteille, retournée et placée sur le compartiment inférieur.Elle sert de stockage pour le parachute. Elle est reliée à la fusée par un fil afin d'assurer la libération du parachute lors de la séparation.

Problème identifié : les essais initiaux ont montré que la coiffe frottait trop contre la structure, ce qui pouvait retarder ou empêcher son détachement.

Amélioration apportée : le bas de la coiffe a été découpé en fines languettes (« floches »), ce qui réduit la surface de contact et les frottements.

Résultat : la séparation devient plus fluide, augmentant la probabilité d'ouverture du parachute et améliorant la fiabilité du système de récupération.

3.4 Coefficient de frottement

Objectif : déterminer le coefficient de frottement de la fusée afin de l'utiliser dans nos équations de prédictions du comportement de la fusée.

Résultat : le coefficient que l'on a obtenu (voir expérience correspondante dans le cahier d'expériences) vaut 0,53 ce qui est assez élevé mais tout de même raisonable, les valeurs typiques se situant entre 0.3 et 0.6.

3.5 Test de poussée

Objectif : comparer les résultats obtenus lors du test de poussée avec les résultats obtenus par résolution numérique du modèle théorique afin de tester la cohérence de notre modèle numérique (voir expérience correspondante dans le cahier d'expérience).

Résultat : bien qu'il y ait quelques différences en terme de valeurs, le fait que les deux courbes aient la même allure nous pousse à penser que notre modèle informatique est assez bien construit.

Conclusion et améliorations : notre objectif au deuxième quadrimestre sera d'étendre notre modèle prédictoire à l'ensemble du vol de la fusée ainsi que de réduire, si possible, les écarts de valeurs entre ce que nous fournit le modèle et la pratique.

4 Présentation du prototype et des codes

4.1 Prototype final

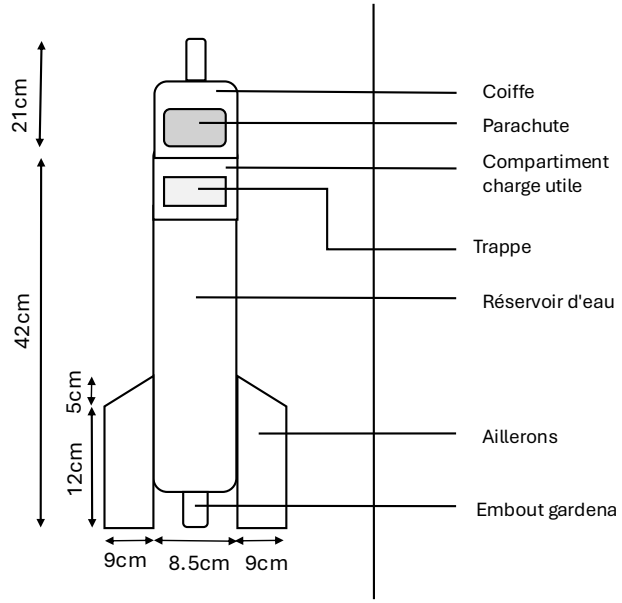

FIGURE 4.1: Schéma des différents composants de la fusée

Les divers composants de la fusée sont illustrés sur le schéma ci-dessus ainsi que leurs côtes correspondantes. La coiffe est posée sur le haut de la fusée et est attachée en un seul point, ce qui cause son ouverture lors de la retombée de la fusée, déployant ainsi le parachute. La trappe du compartiment de la charge utile est directement découpée dans le morceau de bouteille servant de compartiment, la flexibilité du matériau est donc exploitée en guise de charnière. Les quatres ailerons de la fusée sont connectés par deux anneaux placés à leur deux extrémités respectives dont le diamètre intérieur est légerement plus grand que celui de la bouteille à cet endroit.

4.2 Code Python

4.2.1 Objectif du code

Une simulation numérique a été développée en langage Python afin de prédire les performances théoriques du modèle. Cela permettera également, par la suite, de déterminer les paramètres de lancement optimaux afin d'obtenir les résultats voulus. Le code python permet principalement de calculer l'évolution temporelle de trois variables clés : l'altitude (z), la vitesse verticale(v) et le volume d'eau restant dans le réservoir(V_{eau}) pendant la phase de montée de la fusée. Le code complet peut être retrouvé en annexe.5.1

4.2.2 Equations résolues par le code

Le code traduit numériquement le modèle théorique présenté à la section "Recherche théorique" Il résout spécifiquement :

1. L'équation de la Force Poids de la fusée :

$$F_g = m_{total} \cdot g$$

2.12

2. L'équation de la force de frottements :

$$F_{fr} = \frac{1}{2}\rho_{air} \cdot A_b \cdot C_d \cdot v^2$$

2.11

3. L'équation de la vitesse de la fusée (Bernouilli) :

$$v_{eau} = \sqrt{\frac{2(P(t) - P_{atm} + \rho_{eau} \cdot g \cdot h(t))}{\rho_{eau}}}$$

1.10

4. L'équation de la Force de poussée :

$$F_{poussee} = 2(P(t) - P_{atm} + \rho_{eau} \cdot g \cdot h(t)) \cdot A_{eau}$$

1.16

4.2.3 Structure du code

Le code est structuré en 5 parties :

1. Initialisation : C'est dans cette partie que l'on fournit à python : les constantes physiques, les cotes 4.1 de la fusee et les paramètres de lancement.
2. Fontion "model" : Il s'agit de la fonction principale qui prend en entrée le temps t et le vecteur d'état $Y = [z, v, v_{eau}]$ et qui renvoie les dérivées temporelles $[\frac{dz}{dt}, \frac{dv}{dt}, \frac{V_{eau}}{dt}]$. Elle contient également la logique pour passer entre la phase de propulsion par l'eau et la phase ballistique (la phase de propulsion par l'air est pour l'instant ignorée).
3. Gestion des évenements : 2 fonctions sont définies : la première, "plus d eau" qui sert à afficher une droite verticale (d'abscisse : $t = t_{fineau}$) en pointillés rouges lorsque l'eau de la bouteille est entièrement évacuée et également à afficher cette valeur temporelle et la deuxième, "arrivee sommet" qui est l'évenement d'arrêt de la simulation lorsque la fusée arrive à sa hauteur maximale.
4. Resolution numérique : la résolutio de méthode Runge-Kutta d'ordre 4 (RK45) qui adapte automatiquement le pas de temps pour garantir des résultats précis, se fait grâce à la fonction "solve ivp" de la bybliothèque scientifique "scipy".
5. Visualisation : Les résultats sont finalement tracés sous forme de graphique via "Matplotlib"

5 Annexes

5.1 Code Python

Listing 5.1: Résolution numérique du modèle

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
from math import pi, sqrt

#1. Constantes et Paramètres (Unité SI : m, kg, s, Pa)

# Constantes physiques
rho_eau = 998.2        # Masse volumique eau (kg/m^3)
rho_air_atm = 1.204    # Masse volumique air à P_atm (kg/m^3)
P_atm = 101300       # Pression atmosphérique (Pa)
g = 9.81               # Accélération gravitationnelle (m/s^2)
gamma = 1.4            # Coefficient adiabatique pour l'air (diatomique)

# Paramètres de la fusée
Vol_b = 1.5 / 1000     # Volume total de la bouteille (1.5 L -> m^3)
A_b = pi * (0.082 / 2)**2 # Aire de la section de la bouteille (m^2)
A_eau = pi * (0.009 / 2)**2 # Aire de l'orifice de sortie (m^2)

m_fusee_vide = 0.15    # Masse de la fusée vide (kg)
Cd = 0.53               # Coefficient de traînée (exemple)

# Conditions initiales
P_init_relative = 5 * P_atm    # Pression initiale relative (4 bar)
P_init = P_init_relative + P_atm # Pression initiale absolue (Pa)
```

```
Vol_eau_init = 0.5 / 1000 # Volume initial d'eau (0.5 L -> m^3)

# Calculs des conditions initiales
Vol_air_init = Vol_b - Vol_eau_init #volume de la bouteille - volume de l'
    eau
rho_air_init = rho_air_atm * (P_init / P_atm)**(1/gamma) #Loi adiabatique
m_air_init = rho_air_init * Vol_air_init  #masse volumiue * volume

print(f"Pression initiale: {P_init/P_atm:.1f} bar abs.")
print(f"Volume air initial: {Vol_air_init*1000:.1f} L")
print(f"Masse air initiale: {m_air_init:.3f} kg")
print(f"Masse eau initiale: {rho_eau * Vol_eau_init:.3f} kg")

#2. fonction model pour solve_ivp

def model(t, Y):
    """
    Calcule les dérivées de l'état Y = [z, v, V_eau] à l'instant t.
    Y[0] = z (altitude)
    Y[1] = v (vitesse de la fusée)
    Y[2] = V_eau (volume d'eau restant)
    """
    z, v, V_eau = Y

    #Phase 1: Propulsion par Eau
    if V_eau > 0:
        # Calculer les variables dépendantes de V_eau
        Vol_air = Vol_b - V_eau

        #Pression de l'air (Loi de Poisson - Adiabatique)
        P_t = P_init * (Vol_air_init / Vol_air)**gamma #V_0 c'est le volume
            de l'air normalement

        #Hauteur d'eau (modèle cylindrique)
        h_t = V_eau / A_b

        #Pression manométrique à la sortie
        P_gauge_sortie = (P_t + rho_eau * g * h_t) - P_atm

        #Si la pression interne n'est plus suffisante
```

```
        if P_gauge_sortie <= 0:
            V_eau = 0 # On force la fin de la phase eau
            F_poussee = 0
            v_eau = 0
        else:
            #Vitesse d'éjection de l'eau (Bernoulli)
            v_eau = sqrt(2 * P_gauge_sortie / rho_eau)

            #Poussée
            F_poussee = 2 * A_eau * P_gauge_sortie

    #Phase 2: Balistique (plus d'eau)
    else:
        # Plus d'eau, plus de poussée, débit d'eau nul
        V_eau = 0 # Assurer que V_eau ne devienne pas négatif
        F_poussee = 0
        v_eau = 0

    #Forces s'appliquant dans les deux phases

    #Masse totale instantanée
    m_eau = rho_eau * V_eau # m_eau devient 0 en phase 2
    m_total = m_fusee_vide + m_eau + m_air_init # On suppose la masse d'air
        constante

    #Force de gravité
    F_gravite = m_total * g

    #Force de frottements (drag)
    #Note : v peut être négatif (redescente), F_frottements doit s'opposer
    F_frottements = 0.5 * rho_air_atm * (v**2) * A_b * Cd * np.sign(v) #np.
        sign(v) sert à trouver le signe de v, cela est utile pour savoir le
        signe des frottements

    #Somme forces
    F_somme = F_poussee - F_gravite - F_frottements

    #Calcul des dérivées
    dz_dt = v
    dv_dt = F_somme / m_total
```

```
    dV_eau_dt = -v_eau * A_eau

    return [dz_dt, dv_dt, dV_eau_dt]

#3. Définition des événements

def arrivee_sommet(t, Y):
    """
    événement pour stop la simulation lorsuq'elle arrive au sommet / sa vitesse devient nulle (change de signe)
    """
    return Y[1] # Renvoie vitesse fusee. L'événement se déclenche quand vitesse fusée = 0.

arrivee_sommet.terminal = True # Arrête la simulation
arrivee_sommet.direction = -1 # Se déclenche quand vitesse_fusee passe de >0 à <0

def plus_d_eau(t, Y):
    """événement pour voir quand la phase de propulsion s'arrête"""

    return Y[2] #pareil avec V_eau
plus_d_eau.terminal = False #n'arrête pas la simulation
plus_d_eau.direction = -1

#4. Résolution de l'Équation Différentielle

#État initial
z0 = 0.0
v0 = 0.0
V_eau0 = Vol_eau_init
Y0 = [z0, v0, V_eau0]

#Intervalle de temps
t_span = [0, 10] # Simuler pour 10 secondes, Sert aussi à dire que t commence à 0

#Appel au solveur
sol = solve_ivp(
    model,
```

```
    t_span,
    Y0,
    method='RK45', # Méthode de Runge-Kutta
    dense_output=True, #crée une fonction mathématique lisse et continue qui
        passe par les points calculés par Runge-Kutta
    events=[arrivee_sommet,plus_d_eau] #cherche la racine de la fonction
       plus_d_eau qui se produit lorsqu'il n'y a effectivement plus d'eau
       dans la bouteille car la fonction fin de l'eau renvoie simplement la
       valeur du volume d'eau présent dans la bouteille, pareil pour la
       vitesse mais stoppe la simulation en plus de cela
)

#Récupération des résultats
t = sol.t
z = sol.y[0]
v = sol.y[1]
V_eau = sol.y[2]

#Temps de fin de propulsion eau
if sol.t_events[1].size > 0: #event plus d'eau
    t_fin_eau = sol.t_events[1][0]
    print(f"\nFin de la propulsion par eau à t = {t_fin_eau:.3f} s")
else:
    t_fin_eau = t[-1]
    print("\nLa simulation s'est terminée avant la fin de l'eau.")

if sol.t_events[0].size > 0 : #event apogée
    t_apogee = sol.t_events[0][0]
    z_max = sol.y_events [0][0][0] #altitude au moment de l'évenement
    print(f"> Apogée atteinte à : {t_apogee:.3f} s")
    print(f"> Altitude maximale : {z_max:.2f} m")
else:
    print("> La simulation s'est arrêtée avant l'apogée.")



#5. Visualisation des Résultats

fig, axs = plt.subplots(4, 1, figsize=(10, 15), sharex=True)

```

```
172 #Altitude
173 axs[0].plot(t, z, label='Altitude (z)')
174 axs[0].set_ylabel('Altitude (m)')
175 axs[0].set_title('Simulation de la Fusée à Eau (Phase Propulsée)')
176 axs[0].axvline(t_fin_eau, color='r', linestyle='--', label="Fin de l'eau")
177 axs[0].grid(True)
178 if t_fin_eau: axs[0].axvline(t_fin_eau, color='r', linestyle='--') #
    continuer d'afficher une ligne verticale lorsque l'eau se termine
179 axs[0].legend()
180
181 #Vitesse
182 axs[1].plot(t, v, label='Vitesse (v)')
183 axs[1].set_ylabel('Vitesse (m/s)')
184 axs[1].axvline(t_fin_eau, color='r', linestyle='--', label="Fin de l'eau")
185 axs[1].grid(True)
186 if t_fin_eau: axs[1].axvline(t_fin_eau, color='r', linestyle='--')
187 axs[1].legend()
188
189 #Volume d'eau
190 axs[2].plot(t, V_eau * 1000, label='Volume d\'eau (V_eau)')
191 axs[2].set_ylabel('Volume d\'eau (L)')
192 axs[2].set_xlabel('Temps (s)')
193 axs[2].axvline(t_fin_eau, color='r', linestyle='--', label="Fin de l'eau")
194 axs[2].grid(True)
195 if t_fin_eau: axs[2].axvline(t_fin_eau, color='r', linestyle='--')
196 axs[2].legend()
197
198 #Pression à l'intérieur de la bouteille
199
200 #Calcul a posteriori de la pression
201 #On réutilise la loi adiabatique sur le tableau complet V_eau
202 Vol_air_instantané = Vol_b - V_eau
203 P_absolue_vec = P_init * (Vol_air_init / Vol_air_instantané)**gamma
204 P_gauge = (P_absolue_vec - P_atm) #transformation en pression relative
205
206
207 if t_fin_eau: #forcer la pression à zero lorsqu'il n'y a plus d'eau
208     indices_apres_eau = t > t_fin_eau
209     P_gauge[indices_apres_eau] = 0
210
```

```
211
212 axs[3].plot(t,P_gauge, label = 'Pression intérieur bouteille')
213 axs[3].set_ylabel("Pression (Pa)")
214 axs[3].set_xlabel("Temps (s)")
215 axs[3].axvline(t_fin_eau,color = 'r', linestyle = '--', label = "Fin de l'
    eau")
216 axs[3].grid(True)
217 if t_fin_eau: axs[3].axvline(t_fin_eau, color='r', linestyle='--')
218 axs[3].legend()
219
220 plt.tight_layout()
221 plt.show()
```

Bibliographie

[] *L'équation de Tsiolkovski* (s. d.). URL : https://www.numworks.com/fr/professeurs/activites-pedagogiques/terminale/rocket-equation/teacher.pdf.

[Nak24] NAKKA, Richard (2024). *Introduction to rocket design*. URL : https://www.nakka-rocketry.net/RD_fin.htm.

[] *Water rocket calculations* (s. d.). URL : https://www.et.byu.edu/~wheeler/benchtop/flight.php.

[Whe02] WHEELER, Dean (2002). *Modeling the thrust phase*. URL : https://www.et.byu.edu/~wheeler/benchtop/pix/thrust_eqns.pdf.

[Whe] — (s. d.). *Water-rocket flight equations*. URL : https://www.et.byu.edu/~wheeler/benchtop/flight.php.