RAPPORT DE VALIDATION : BUREAU D'ÉTUDES

Balistique & Trajectoire

Calculs Scientifiques & Programmation

Quentin Bergé
Marc Ferrière
ENSEEIHT

Fevrier 2019

Table des matières

1 Introduction

Il est question ici de créer un programme pour effectuer le calcul de trajectoire d'un objet en chute libre et propulsé selon l'enoncé proposé. Dans ce rapport nous aborderons tout d'abord la modélisation du problème, puis nous nous focaliserons sur l'architecture du programme pour enfin aborder la validation.

2 Modélisation

2.1 Chute Libre

Si on effectue le bilan des forces sur un objet de masse m possédant une vitesse initiale v_0 formant un angle α avec l'axe des abscisses à une altitude initiale h :

$$\Sigma\vec{F} = m \times \vec{a}$$

Ici $\vec{F}$ se limitera à $\vec{P} = m\vec{g}$ d'où $g = a$

2.1.1 Solutions Analytiques

On peut alors obtenir les équations analytiques avec $x(t)$ et $z(t)$ qui nous serviront de références :

$$x(t) = v_0 \cos(\alpha)t$$
$$z(t) = v_0 \sin(\alpha)t - \frac{gt^2}{2} + h$$

On peut aussi obtenir les équations qui nous donneront la portée et le temps :

$$portee = \frac{v_0}{g}\cos(\alpha)\sqrt{v_0 \sin(\alpha) + (v_0 \sin(\alpha))^2 + 2gh}$$
$$t_l = \frac{portee}{v_0 \cos(\alpha)}$$

2.1.2 Modélisation Euler

Le schéma d'Euler d'ordre 1 s'écrit :

$$y_{k+1} = y_k + \Delta t \frac{df}{dt}(t_k, y_k)$$

On utilise un tableau de 4 colonnes noté y :

$$\begin{cases} y(1,i) = x(i) \\ y(2,i) = z(i) \\ y(3,i) = v_x(i) \\ y(4,i) = v_z(i) \end{cases}$$

$$\begin{cases} y(1,i) = x(i) \\ y(2,i) = z(i) \\ y(3,i) = v_x(i) \\ y(4,i) = v_z(i) \end{cases} \iff \begin{cases} \frac{dx}{dt} = v_x \\ \frac{dz}{dt} = v_z \\ \frac{dv_x}{dt} = 0 \\ \frac{dv_z}{dt} = -g \end{cases}$$

Ainsi on peut modéliser notre trajectoire de chute libre par un système du type :

$$\begin{cases} y(3,i+1) = y(3,i) \\ y(4,i+1) = y(4,i) - dt \times g \\ y(1,i+1) = y(1,i) + dt \times y(3,i) \\ y(2,i+1) = y(2,i) + dt \times y(4,i) \end{cases}$$

A chaque pas de temps , on recalcule les vitesses et la position en x et en z à partir du système précédent.

2.1.3 Modélisation Runge-Kutta d'ordre 4

Le schéma de Runge-Kutta d'ordre 4 :

$$y_{k+1} = y_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k4)$$

Avec :

$$\begin{cases} k_1 = \frac{dy_k}{dt} \\ k_2 = \frac{dy_k}{dt} + \frac{\Delta t}{2} \times k_1 \\ k_3 = \frac{dy_k}{dt} + \frac{\Delta t}{2} \times k_2 \\ k_4 = \frac{dy_k}{dt} + \Delta t \times k_3 \end{cases}$$

On utilise les dérivées de chute libre de la méthode Euler 2.1.2 en appliquant la méthode Runge-Kutta à l'ordre 4 (2.1.3).

2.2 Objet Portant Propulsé

Maintenant notre masse est propulsé avec une force aérodynamique et on prend en compte la portance et la traînée dans nos calculs.

2.2.1 Méthode Euler

On reprend le même schéma d'Euler à l'ordre 1 (2.1.2) que pour la trajectoire en chute libre mais en intégrant les forces de portance L, de traînée D et d'aérodynamique F_0

$$\frac{dx}{dt} = v_x$$

$$\frac{dz}{dt} = v_z$$

$$\frac{dv_x}{dt} = \frac{F_0\cos(\theta) - L\sin(\theta) - D\cos(\theta)}{m}$$

$$\frac{dv_z}{dt} = \frac{-g + F_0 \sin(\theta) + L\cos(\theta) - D\sin(\theta)}{m}$$

Avec θ l'angle entre entre les vecteurs vitesses v_x et v_z :

$$\theta = \arctan(\frac{v_z}{v_x})$$

2.2.2 Méthode Runge-Kutta d'ordre 4

On reprend le schéma Runge-Kutta d'ordre 4 (2.1.3) en modélisant les nouvelles dérivées vu en 2.2.1.

3 Architecture

Cette partie va détailler l'architecture du programme.

3.1 Contenu

Le programme est placé dans le dossier `Fortran` qui comprend :
— un fichier `mod_balistique.f90`
— un fichier de subroutines `subroutines.f90`
— le programme `main.f90`
— le fichier `Makefile` permettant la compilation

Le dossier `RUN` comprend :
— les fichier de sorties de forme `BE_Methode_Modele_npt_XXXX_.out` et `Paramétrisation_Alpha_Methode_Modele.out`
— le fichier d'entrée `balistique.in`
— l'exécutable `main.bin`

Dans le dossier `Validation` sont compris :
— les résultats du programme pour les différents modèles avec les deux modélisation
— les graphiques de résultats

Le dossier `Rapport` comprend les ressources pour le rapport LaTeXainsi que le rapport en lui-même.

3.2 Algorithmie

Voici un algorigramme général présnetant le fonctionnement du programme :

On détaille la procédure pour RK4 :

On a créer une subroutine nommée `RK4` qui a pour paramètres d'entrées :
— dérivée de la fonction
— n : numéro du vecteur qu'on veut calculer

Elle calcule la valeur du vecteur y_{k+1} selon le schéma vu en 2.1.3.

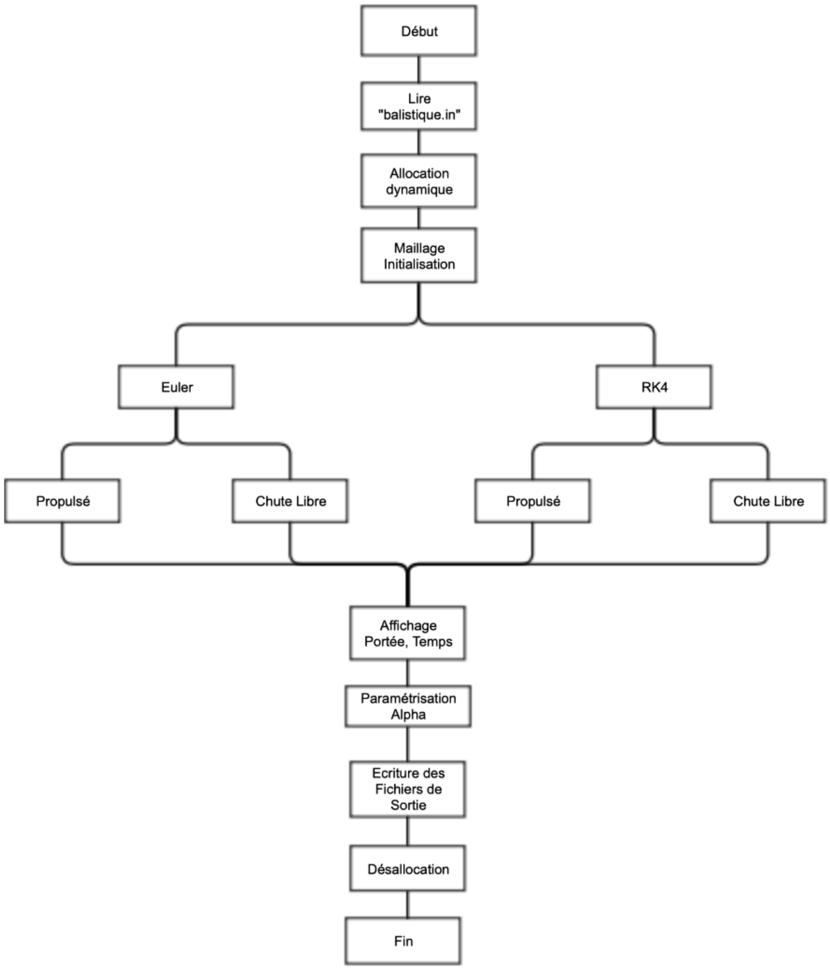

FIGURE 1 – Algorigramme

Ensuite on appelle cette subroutine dans les 2 cas (Chute libre ou Runge-Katta d'ordre 4) et selon les différentes forces qui s'appliquent, calcule les vecteurs positions et vitesses en x et en z.

Pour la variation de l'angle α, la subroutine `parametrisation_alpha` fait varier l'angle alpha de 25 ° jusqu'à 75 ° par pas de 5 °.
Ensuite cette subroutine reprend l'architecture du programme principal `main.f90` pour calculer les trajectoires de chaque angle *alpha*.
On affiche ensuite la portée et le temps de chute pour tous les angles.

4 Validation

On effectue la validation selon la solution analytique pour le cas de la chute libre pour les deux méthodes.

Tout d'abord selon la méthode d'Euler.

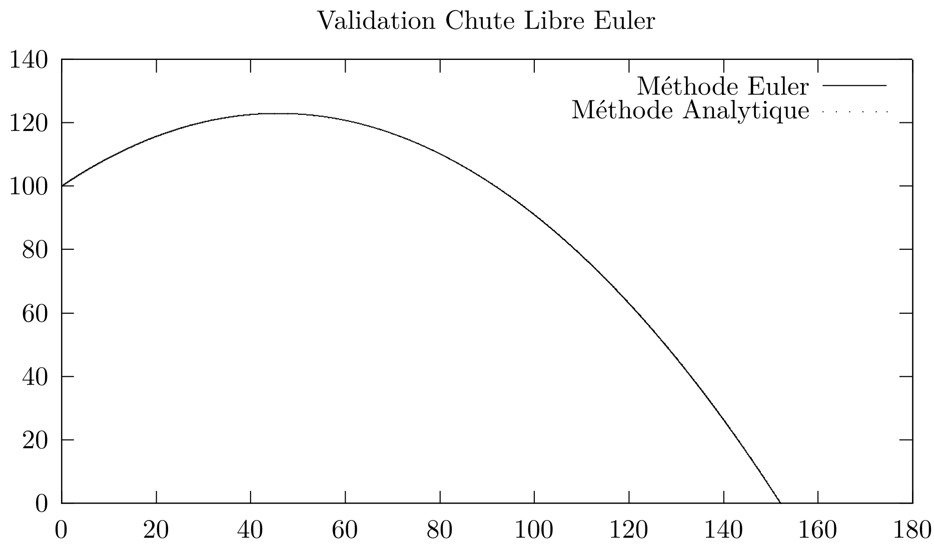

Figure 2 – Validation de la Chute Libre selon la méthode d'Euler avec la méthode Analytique

On observe la parfaite superposition entre les deux courbes, ainsi on peut valider nos résultats avec la résolution de la méthode d'Euler.

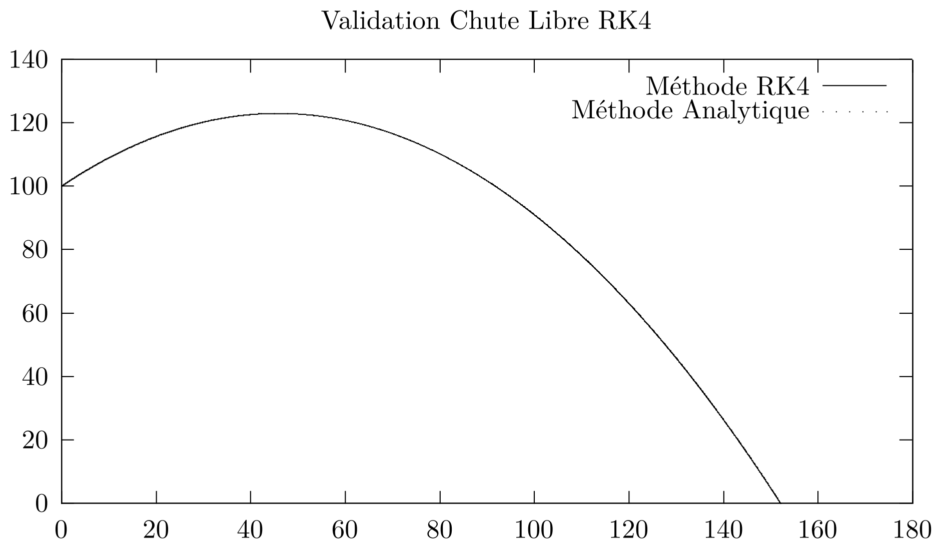

FIGURE 3 – Validation de la Chute Libre selon la méthode RK4 avec la méthode Analytique

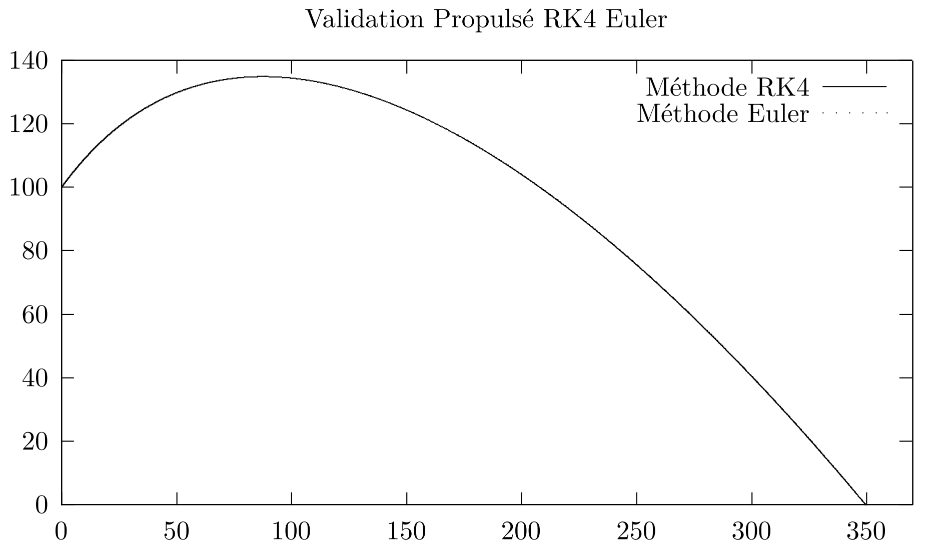

FIGURE 4 – Validation du modèle Propulsé selon la méthode RK4 et la méthode Euler

5 Conclusion