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 Résumé

2.1 Résumé en Anglais

2.2 Résumé en Français

3 Modélisation

3.1 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 uniquement soumis à la gravité :

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

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

3.1.1 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 :

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

3.1.2 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 qui répertorie les positions et les vitesses comme suit, on obtient alors les équations régissant la dynamique de l'objet en chute libre :

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

On peut alors modéliser notre trajectoire de chute libre par un système du type :

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

Avec dt le pas de temps utilisé définit comme :

$$dt = \frac{nombredepoints}{tempssimulation}$$

A chaque itération, on recalcule les vitesses v_x et v_z, ainsi que les positions en x et en z à partir du système précédent.

3.1.3 Modélisation Runge-Kutta d'ordre 4

Le schéma de Runge-Kutta d'ordre 4 est construit comme ceci :

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

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 utilisera les dérivées de la modélisation chute libre de la méthode d'Euler 3.1.2 en appliquant la méthode Runge-Kutta à l'ordre 4.

3.2 Objet Portant Propulsé

On étudiera ici notre objet de masse m possédant une vitesse initiale v_0 formant un angle α avec l'axe des abscisses à une altitude initiale h, sera soumis à une force de propulsion F_0, une force de portance L et une force de trainée D.

3.2.1 Méthode Euler

On reprend le même schéma d'Euler à l'ordre 1 (cf. 3.1.2) en prenant en compte les forces de portance L, de traînée D et de propulsion F_0 :

$$\begin{cases} \frac{dx}{dt} = v_x \\ \frac{dz}{dt} = v_z \\ \frac{dv_x}{dt} = \frac{F_0\cos(\theta) - L\sin(\theta) - D\cos(\theta)}{m} \\ \frac{dv_z}{dt} = \frac{-g + F_0\sin(\theta) + L\cos(\theta) - D\sin(\theta)}{m} \end{cases}$$

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

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

3.2.2 Méthode Runge-Kutta d'ordre 4

On reprend le schéma Runge-Kutta d'ordre 4 (cf 3.1.3) en modélisant les nouvelles dérivées vues en 3.2.1.

4 Architecture

Nous détaillerons ici le contenu de l'archive ainsi que l'algorithme du programme.

4.1 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.

4.2 Algorithmie

Voici l'algorigramme général présentant le fonctionnement du programme.

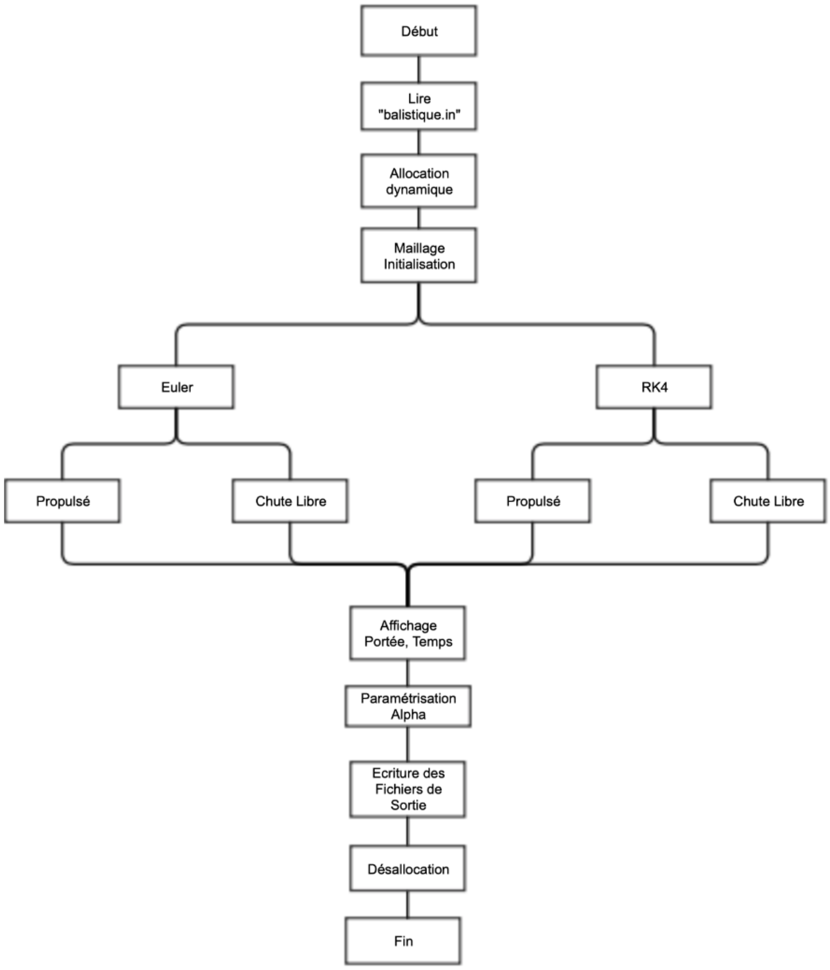

FIGURE 1 – Algorigramme général du programme

4.2.1 Procédure RK4

Détaillons la procédure pour Runge-Kutta d'ordre 4.

On a créé une subroutine nommée `rk4` qui a pour paramètres d'entrées :
— `derivee` : dérivée de la fonction
— `n` : numéro du vecteur que l'on veut calculer
Elle calcule la valeur du vecteur y_{k+1} selon le schéma vu en 3.1.3.

On appelle cette subroutine dans les 2 cas pour calculer les positions et les vitesses en x et en z :
— Chute Libre
— Propulsé

4.2.2 Paramétrisation d'Angle

Pour la variation de l'angle α, Dans la subroutine `parametrisation_alpha`, on modifie la variable globale de l'angle α de 25 $^\circ$ jusqu'à 75 $^\circ$ par pas de 5 $^\circ$. On appelera alors les différentes subroutines présentes dans dans `main.f90` en reprenant l'architecture du programme pour calculer les trajectoires de chaque angle α. On affiche ensuite la portée et le temps de chute pour tous les angles.

5 Validation

On effectue la validation selon la solution analytique pour le cas de la chute libre pour les deux méthodes. Les graphiques sont réalisés dans le LATEXà l'aide de gnuplot.

5.1 Trajectoires

5.1.1 Modèle Chute Libre

Tout d'abord selon la méthode d'Euler.

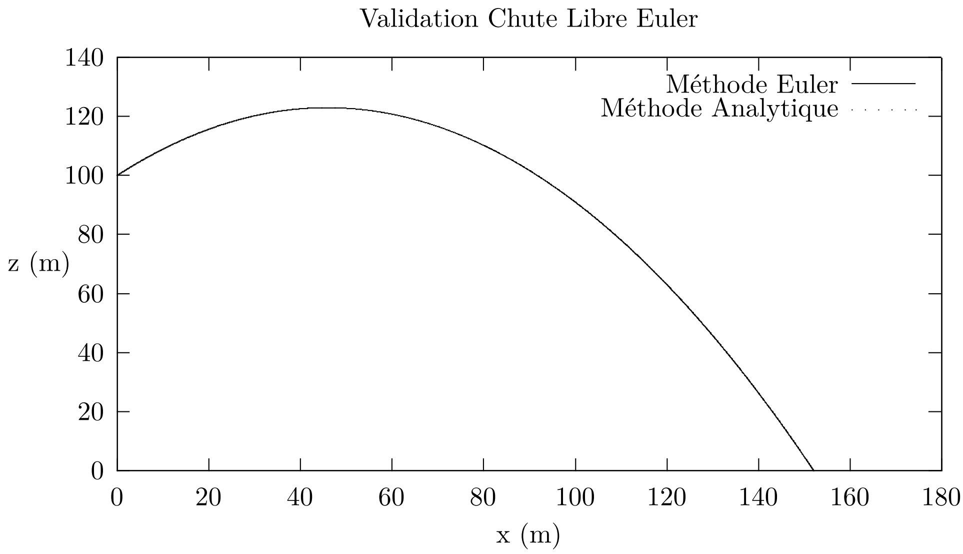

FIGURE 2 – Validation de la trajectoire en Chute Libre selon la méthode d'Euler avec la méthode Analytique

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

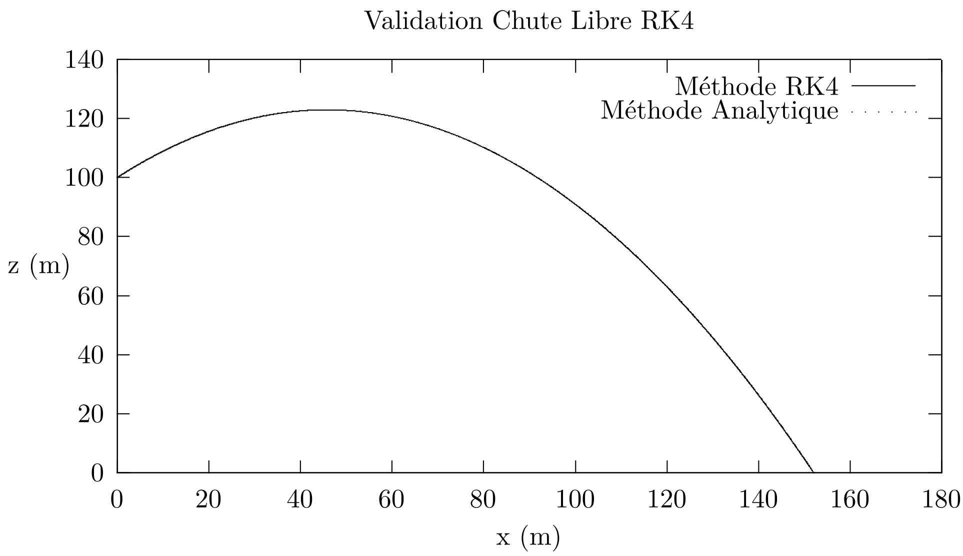

FIGURE 3 – Validation de la trajectoire en Chute Libre selon la méthode RK4 avec la méthode Analytique

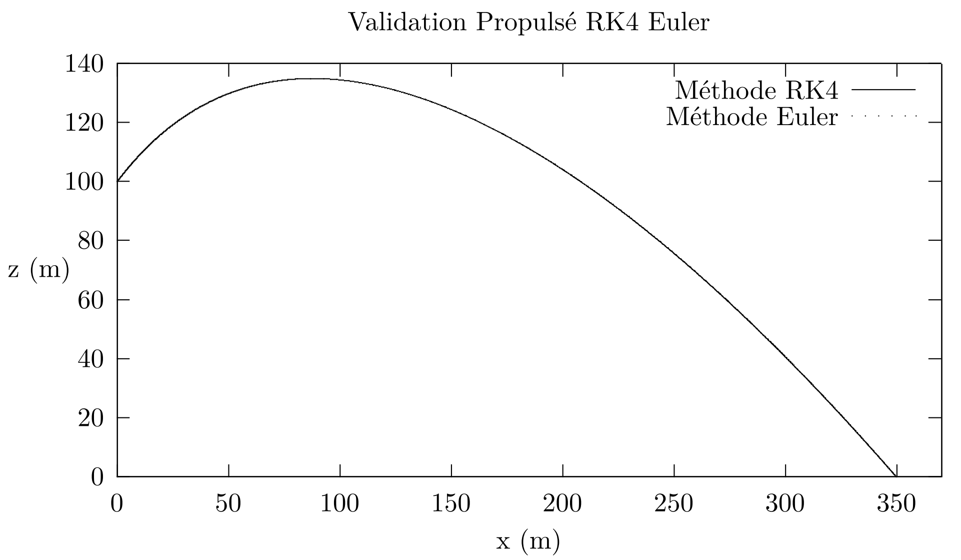

FIGURE 4 – Validation du modèle Propulsé selon la méthode RK4 et la méthode Euler

5.1.2 Modèle Propulsé

5.2 Paramétrisation d'Angle

5.2.1 Modèle Chute Libre

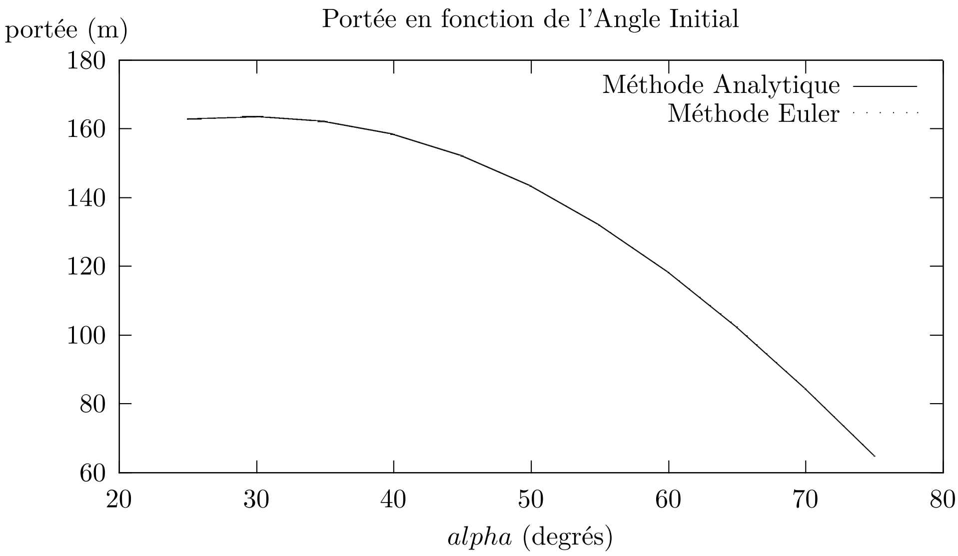

FIGURE 5 – Validation paramétrisation d'alpha avec la portée en Chute Libre selon la méthode Euler avec la méthode Analytique

5.2.2 Modèle Propulsé

Passons maintenant au modèle Propulsé.

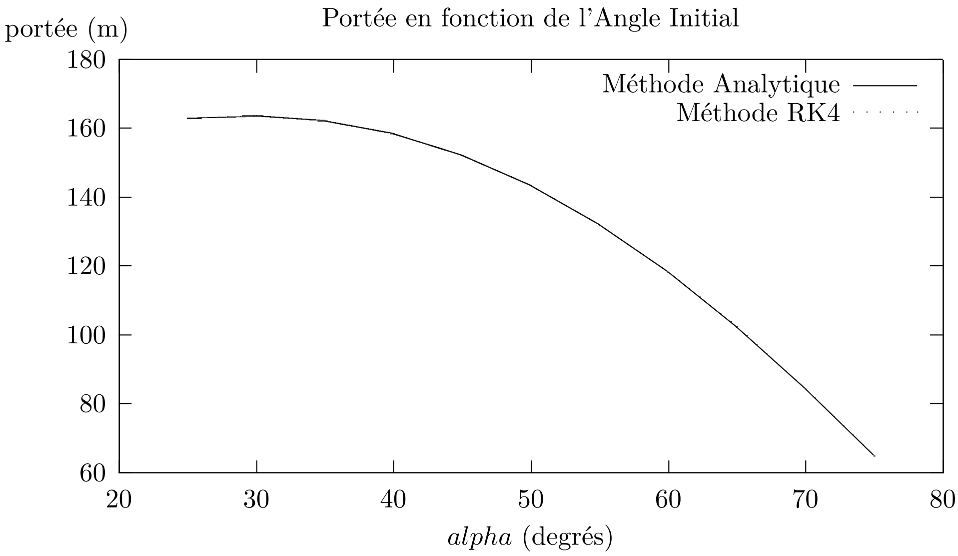

FIGURE 6 – Validation paramétrisation d'alpha avec la portée en Chute Libre selon la méthode RK4 avec la méthode Analytique

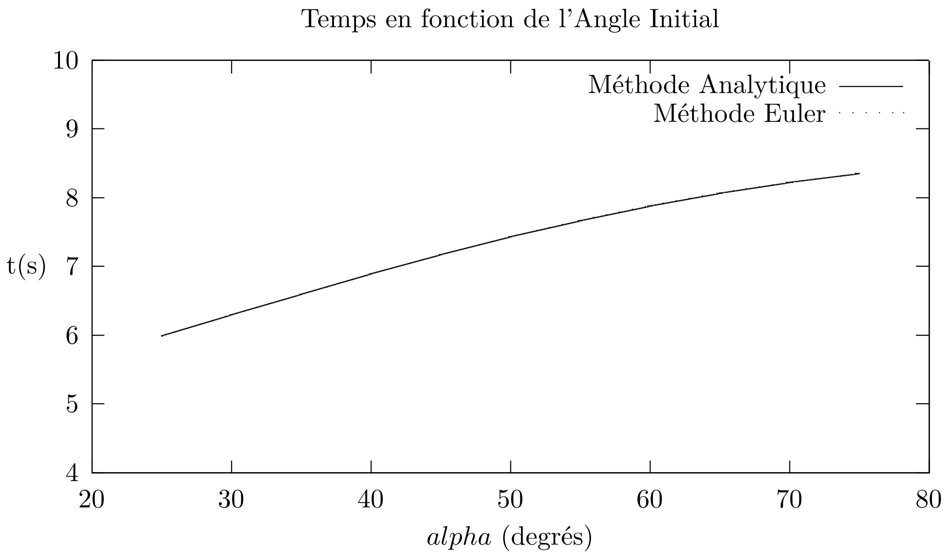

FIGURE 7 – Validation paramétrisation d'alpha avec le temps en Chute Libre selon la méthode Euler avec la méthode Analytique

6 Conclusion

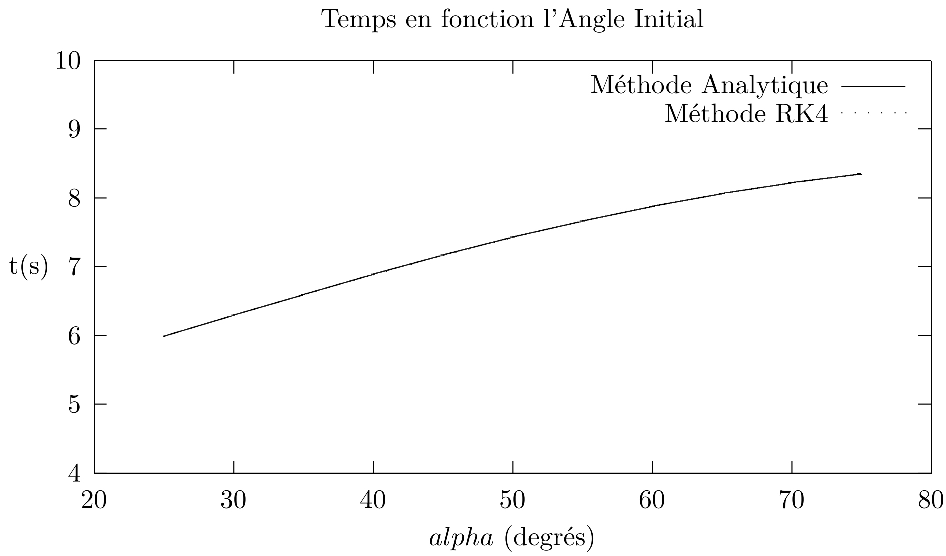

FIGURE 8 – Validation paramétrisation d'alpha avec le temps en Chute Libre selon la méthode RK4 avec la méthode Analytique

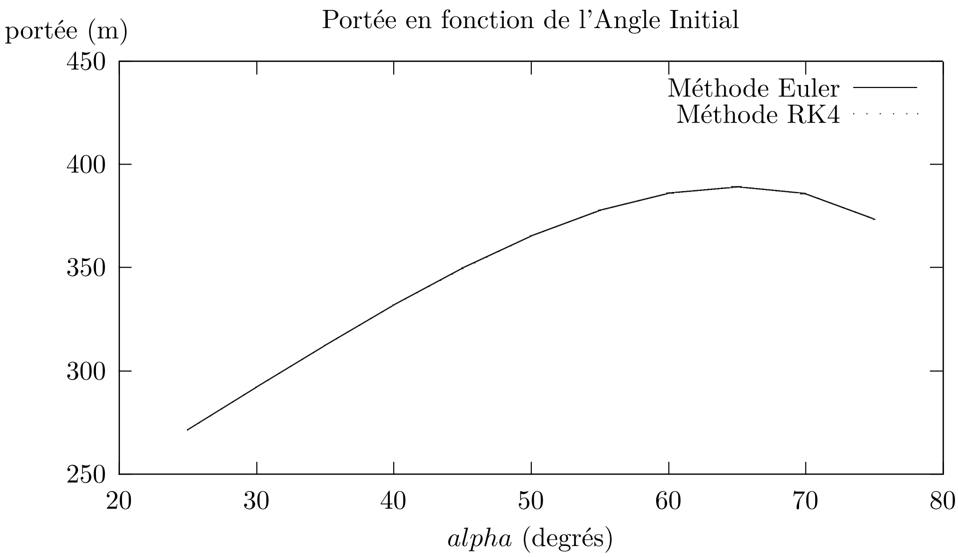

FIGURE 9 – Validation paramétrisation d'alpha avec la portée en Propulsé selon la méthode Euler avec la méthode Analytique

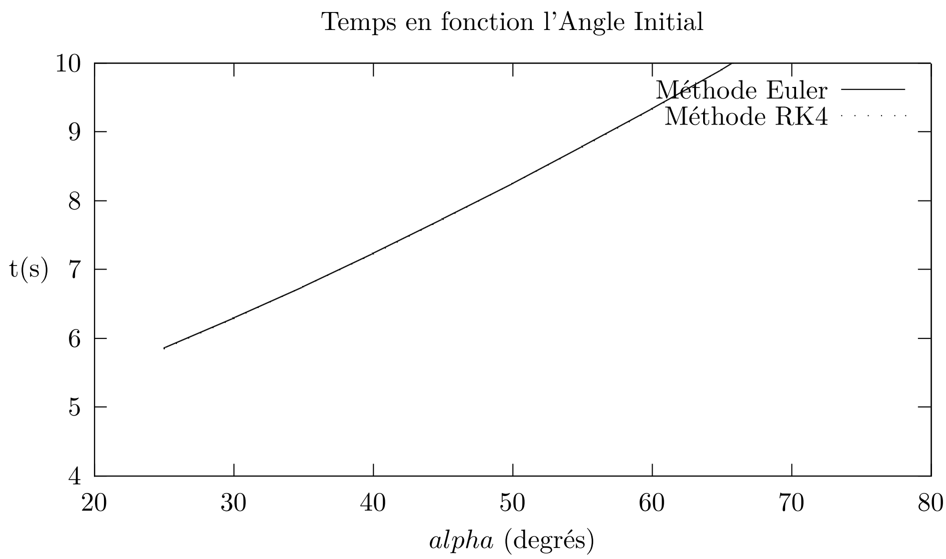

FIGURE 10 – Validation paramétrisation d'alpha avec le temps en Propulsé selon la méthode RK4 avec la méthode Analytique