Compte rendu TD2 - Equations differentielles ordinaires

Celine Bouby

Compte rendu TD2

Réalisé par Juliette BLUEM & Axel THOUVENIN

20 juin 2021

Table des matières

1 Exercice n°1

1.1 Introduction

La population de renards et de lapins au cours du temps évoluent de la façon suivante :

$$\frac{dr}{dt} = 2r - \alpha r f$$

$$\frac{df}{dt} = -f + \alpha r f$$

r étant le nombre de lapins, f celui de renards et t le temps en années. En effet, le renard est un prédateur du lapin.
Nous allons utiliser cet écosystème pour étudier la fonction ode23 disponible sur Matlab. Nous comparerons ainsi les résultats avec des pas fixes et un pas adaptatif.

Dans toutes nos courbes, la population de lapin sera en bleu, et celle de renards en orange.

1.2 Pas de 1

Nous commençons par tracer la solution de notre système avec un pas fixe de 1.

Dans un premier temps on créé une fonction qui permettra de transformer notre système d'équations. L'objectif est de contenir nos deux fonctions dans une seule matrice, On utilise alors les données du système. Notre fonction prend 2 paramètres, le premier qui représente le temps et le second notre matrice de données. On a w[r,f] et wd[rd,fd] qui est notre matrice des dérivées. On utilisera alors "w(1)" pour "r" et w(2) pour "f".

```
function wd = voltb(t,w)
%système ordre 1 donc on peut directement utiliser la formule donnée

alpha = 0.01;
%condition initiales

%définition des systèmes
wd=zeros(size(w)) ;% Calcul des dérivées

wd(1) = 2 * w(1) - alpha * w(1) * w(2);
wd(2) = -w(2) + alpha * w(1) * w(2);

end
```

FIGURE 1.1 – Code de définition de la fonction voltb

Pour ce qui est de l'application de notre fonction, ici dans le cadre d'un pas fixe de 1. Il est nécessaire de modifier la fréquence des points, On utilise la variable "tspan", avec 0 comme point de départ 1 comme pas 25 comme fin.

On utilise part la suite un tableau contenant nos résultats, provenant de la fonction "ode23" qui est, sous Matlab, la méthode d'analyse numérique Runge-Kutta d'ordre 2.

Pour finir on affiche ces résultats, "w(:1)" fait appel a notre premier fonction représentant des lapins et "w(:2)" notre seconde fonction représentant l'évolution des renards.

```
%PAS FIXE
tspan=0 :1 :25 ;
[t,w] = ode23 ('voltb', tspan,w0);%appel la fonction
plot(t,w( :,1),'*-',t,w( :,2),'*-') ;
grid on ;
xlabel('Temps, t, annees');
ylabel('Evolution population lapins / renards');
title ('Evolution population : Volterra') ;
```

FIGURE 1.2 – Code pour tracer la solution avec un pas fixe de 1

Nous obtenons les courbes suivantes :

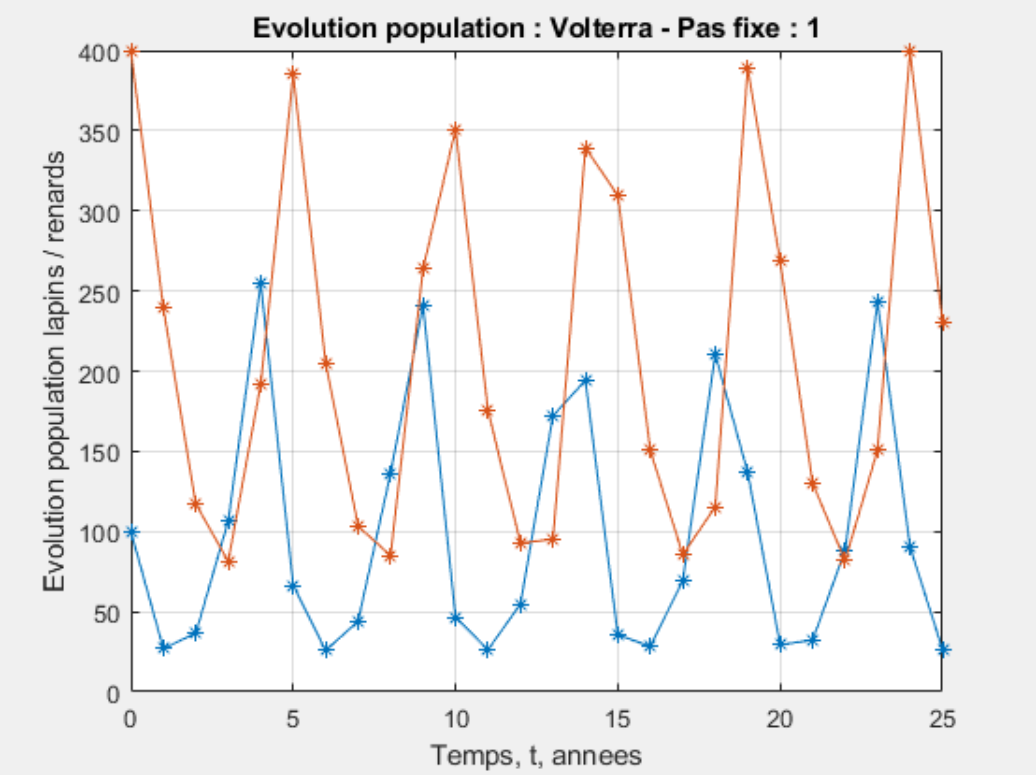

FIGURE 1.3 – Solution avec un pas fixe de 1

Nous voyons que le pas de 1 est bien trop grand, les courbes sont linéaires par morceaux et ces intervalles : trop grands. On peut observer une perte de données, notre solution ne revient jamais à son maximum, elle manque de régularité.
Nous choisissons donc de réduire ce pas pour la suite de notre étude.

1.3 Pas de 0.15 et 0.10

Pour modifier le pas, il suffit de changer le rythme du tspan. Par exemple, pour un pas de 0.15 :

$$tspan = 0 : 2 : 25$$

Voici la courbe de la solution du système avec un pas fixe de 0.15 :

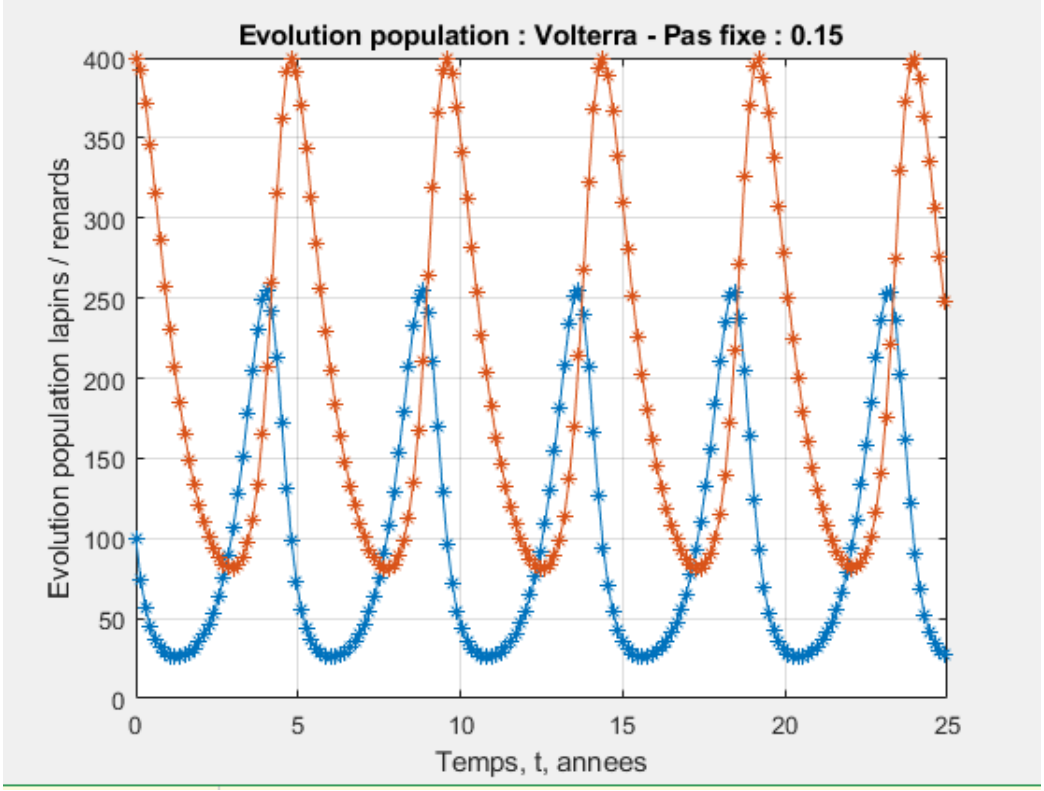

FIGURE 1.4 – Solution avec un pas fixe de 0.15

La courbe est bien plus fluide et consomme 167 points.

Avec un pas de 0.10 :

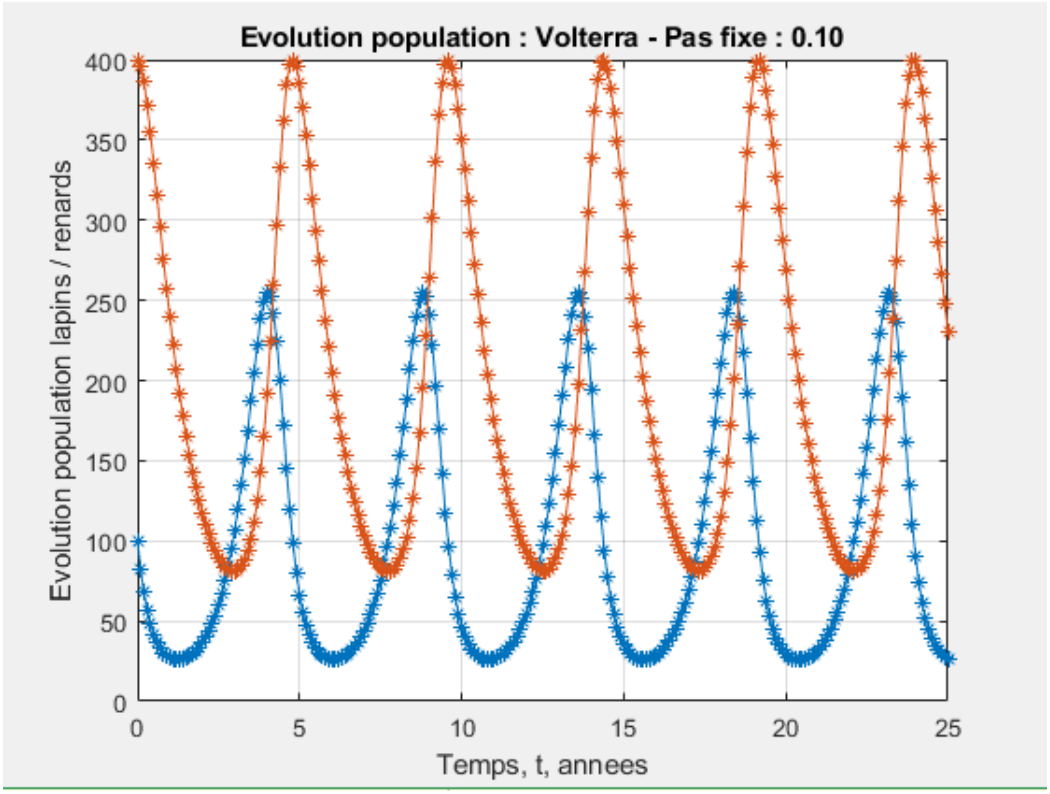

FIGURE 1.5 – Solution avec un pas fixe de 0.1

La courbe est à la limite de la perfection, mais on ne peut la considérer comme optimale car elle est très coûteuse, 251 points.
Pour palier à ce problème, nous allons étudier la solution avec un pas adaptatif.

1.4 Pas adaptatif

Pour ce qui est du code a pas adaptatif, c'est exactement la même chose que le pas fixe. Il faut simplement supprimer notre "tspan". Définir le "t0" comme point de départ et "tf" comme fin. Il suffit par la suite d'appeler cet intervalle dans notre méthode "ode23" qui fera le pas adaptatif automatiquement.

```
%PAS ADAPTATIF
t0    = 0 ; tf=25 ; w0=[100 ;400] ;
[t,w] = ode23 ('voltb', [t0,tf],w0);%appel la fonction
plot(t,w( :,1),'*-',t,w( :,2),'*-') ;
grid on ;
xlabel('Temps, t, annees');
ylabel('Evolution population lapins / renards');
title ('Evolution population : Volterra') ;
```

FIGURE 1.6 – Code pour tracer la solution avec un pas adaptatif

Nous obtenons les courbes suivantes :

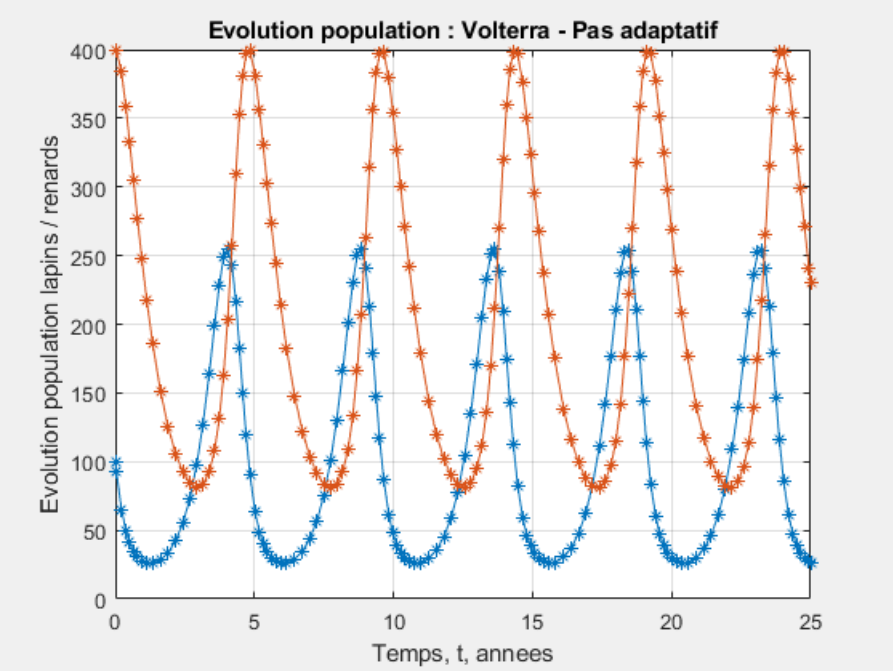

FIGURE 1.7 – Solution avec un pas adaptatif

Le pas adaptatif est le meilleur compromis car la courbe correspond à nos attentes, et elle n'est pas trop coûteuse, 139 points sont nécessaires.

1.5 Conclusion

Via un exemple d'écosystème de Volterra adapté à des renards et des lapins, nous avons pu étudier une méthode graphique de résolution d'équations différentielles ordinaires. Grâce à cette méthode, nous avons comparé les solutions sur trois pas fixes et un pas adaptatif.
Notre bilan est clair, la plus rentable d'un point de vue $\frac{Precision}{CoutDeL'opration}$ est le pas adaptatif.

2 Exercice n°2

2.1 Introduction

Afin de manipuler la résolution d'équations différentielles, nous allons étudier la trajectoire d'une balle de tennis lors d'un lob. Nous supposons pour cela que la balle reste dans un plan (Oxz).
Son équation est donc la suivante.

$$\begin{cases} x''(t) = C_D \alpha v x'(t) + C_M \alpha v z'(t) \\ z''(t) = -g - C_D \alpha v z'(t) + C_M \alpha v x'(t) \\ \alpha = \frac{\pi p d^2}{8m} \\ C_D = 0.508 + \frac{1}{22.503 + 4.196(v/\omega)^2/5} \\ C_M = \frac{1}{2.022 + 0.981(v/\omega)} \end{cases}$$

Nous commençons par ramener notre système d'équations à une forme plus adaptée afin qu'il soit implémenté dans Matlab.

$$\begin{cases} \omega_1'(t) = \omega_2(t) \\ \omega_2'(t) = -C_D \alpha v \omega_2(t) + C_M \alpha v \omega_4(t) \\ \omega_3'(t) = \omega_4(t) \\ \omega_4'(t) = -g - C_D \alpha v \omega_4(t) - C_M \alpha v \omega_2(t) \end{cases}$$

2.2 Trajectoire dans le vide

Si nous effectuons un lob dans le vide, $C_M = C_D = 0$.
Nous devons maintenant résoudre notre système de 4 équations.
Nous commençons pour cela par créer une fonction nommée "tennisbvide" permettant de transformer un système en matrice.

Dans un premier temps nous définissons l'ensemble des variables qui représentent les valeurs numériques de notre système, puis notre α. On implémente notre ω', et nous définissons $v = \sqrt{x_2' + z_2'}$.
Dans ce cas précis nous travaillons dans un espace vide (sans air ni lift). Notre domaine de vitesse est donc nul (C_D et C_M).
Enfin nous implémentons notre système présenté précédemment, qui est adapté au format matriciel de Matlab.

```
function wd = tennisbvide(t,w)
%DONNEES
m = 0.05;%kg
p = 1.29;%kg.m-3
d = 0.063;%m
ws = 20;%rad.s-1
g = 9.81;%m.s-1
alpha = (pi*p*d^2)/(8*m); % = 0.0402

wd=zeros(size(w)) ;% Calcul des dérivées

v = sqrt(w(2)^2+w(4)^2);

CD = 0;
CM = 0;

wd(1) = w(2);
wd(2) = - CD*alpha*v*w(2) + CM*alpha*v*w(4);
wd(3) = w(4);
wd(4) = - g - CD*alpha*v*w(4) - CM*alpha*v*w(2);

end
```

FIGURE 2.1 – Code de définition de la fonction tenisbVide

Dans le code principal on défini la hauteur, la vitesse de la balle et l'angle d'inclinaison. Ces données seront les mêmes tout au long de notre étude.
On défini ensuite notre matrice ω_0 avec les conditions initiales. On affecte par la suite les valeurs de nos fonctions qui ont comme paramètres le temps de départ, le temps de fin et les conditions initiales.

```
%% Exercice 2
clear
clc
h = 2.5;%m
v0 = 25;%m.s-1
teta = pi*15/180;%d°
t0    = 0 ; tf = 2 ; w0=[0;v0*cos(teta);h;v0*sin(teta)];
[t1,w1] = ode23 ('tennisblift', [t0,tf],w0);%appel la fonction;
[t2,w2] = ode23 ('tennisbair', [t0,tf],w0);%appel la fonction;
[t3,w3] = ode23 ('tennisbvide', [t0,tf],w0);%appel la fonction;
plot(t1,w1( :,3),'*-',t2,w2( :,3),'*-',t3,w3( :,3),'*-');

grid on ;
xlabel('Temps, t, secondes');
ylabel('Position z');
legend("Lift","Air","Vide");
title ("Trajectoire d'une balle de tennins lors d'un lob dans le vide") ;
```

FIGURE 2.2 – Code pour tracer la trajectoire de la balle dans le vide

Nous obtenons la courbe suivante :

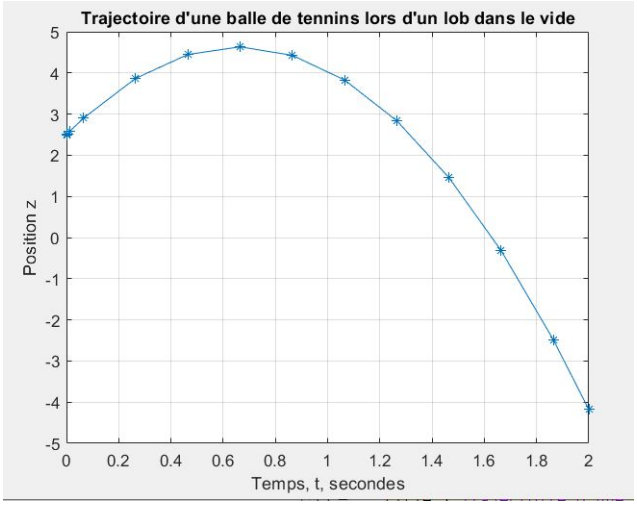

FIGURE 2.3 – Courbe de la trajectoire de la balle dans le vide

Nous pouvons observer que, dans le vide, la balle évolue dans un environnement sans aucune résistance si ce n'est la gravité. On peut espérer que la balle va retomber le plus tardivement possible. Ici la balle rebondit après environ 1.65s (franchissement le l'abscisse 0).

2.3 Trajectoire dans l'air en l'absence du lift

Nous allons maintenant étudier la trajectoire dans l'air, donc $C_M = 0$ et $C_D \neq 0$. De la même façon que dans le vide, nous définissons notre fonction de transformation. Cependant nous ajoutons l'air.

```
function wd = tennisbair(t,w)
%DONNEES
m = 0.05;%kg
p = 1.29;%kg.m-3
d = 0.063;%m
ws = 20;%rad.s-1
g = 9.81;%m.s-1
alpha = (pi*p*d^2)/(8*m); % = 0.0402

wd=zeros(size(w)) ;% Calcul des dérivées

v = sqrt(w(2)^2+w(4)^2);

CD = 0.508 + 1/(22.503+4.196*(v/ws)^(2/5));
CM = 0;


wd(1) = w(2);
wd(2) = - CD*alpha*v*w(2) + CM*alpha*v*w(4);
wd(3) = w(4);
wd(4) = - g - CD*alpha*v*w(4) - CM*alpha*v*w(2);

end
```

FIGURE 2.4 – Code de définition de la fonction tenisbAir

Après un appel identique, nous obtenons la courbe suivante :

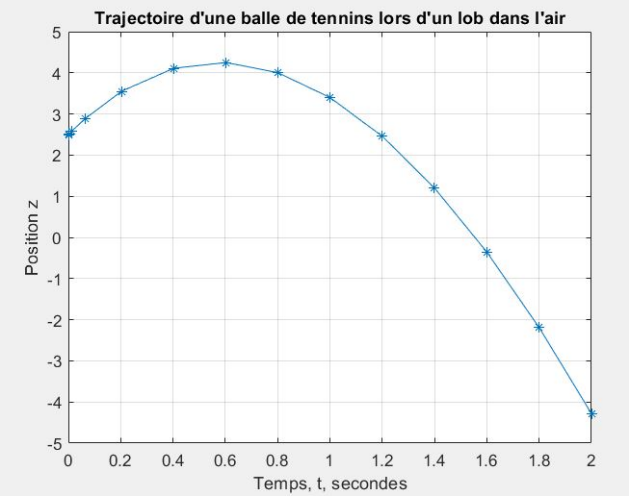

FIGURE 2.5 – Courbe de la trajectoire de la balle dans l'air

Cette fois, l'air est présent. On remarque que la balle fait son premier rebond à 1.58s. Ce qui est plus tôt que la balle dans un environnement vide. Cela s'explique par la résistance apporté sur notre balle. Ce qui impact son avancement.

2.4 Trajectoire dans l'air avec un lift

Enfin, voyons comment se comporte la balle si elle est liftée lors du lob. Dans ce cas $C_D \neq C_M \neq 0$.

Encore une fois, nous créons une fonction de transformation du système d'équations en une matrice.

```
function wd = tennisblift(t,w)
%DONNEES
m = 0.05;%kg
p = 1.29;%kg.m-3
d = 0.063;%m
ws = 20;%rad.s-1
g = 9.81;%m.s-1
alpha = (pi*p*d^2)/(8*m); % = 0.0402

wd=zeros(size(w)) ;% Calcul des dérivées

v = sqrt(w(2)^2+w(4)^2);

CD = 0.508 + 1/(22.503+4.196*(v/ws)^(2/5));
CM = 1/(2.022+0.981*(v/ws));


wd(1) = w(2);
wd(2) = - CD*alpha*v*w(2) + CM*alpha*v*w(4);
wd(3) = w(4);
wd(4) = - g - CD*alpha*v*w(4) - CM*alpha*v*w(2);

end
```

FIGURE 2.6 – Code de définition de la fonction tenisbLift

Et après un appel dans notre programme principal, nous obtenons la courbe suivante.

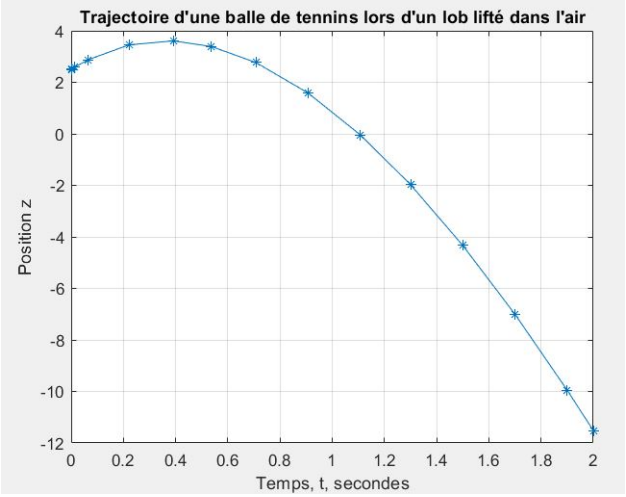

FIGURE 2.7 – Courbe de la trajectoire de la balle liftée dans l'air

On remarque que le premier rebond se situe à environ 1.1s. Ce qui est beaucoup plus tôt que nos deux courbes précédentes. Ce qui est normale. Dans le cas d'un lift la balle prend un effet et tend plus rapidement vers le sol. Ce qui explique la rapidité d'apparition du premier rebond.

2.5 Conclusion

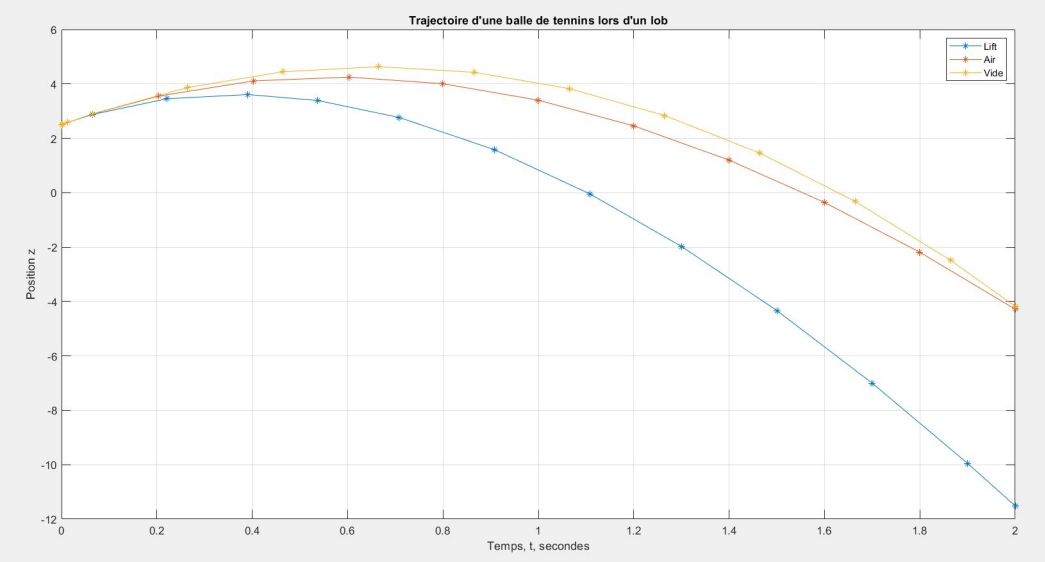

FIGURE 2.8 – Comparaison des différentes trajectoires

La durée du lob de la courbe jaune (vide) est de 1.65s, celle de la courbe orange (air) 1.58s et enfin la courbe bleue (lift) 1.1s. Cela est dû à l'environnement et les différentes résistances ou forces qui sont appliquées à notre balle. Notre système représente parfaitement la trajectoire d'une balle lors d'un lob. Sans aucune résistance la balle prend plus de temps à faire son premier rebond, avec le frottement de l'air elle prend un peu moins de temps, si on ajoute au frottement un effet lifté la balle atteint le sol bien plus rapidement.
Via cet exemple, nous avons clairement manipulé et développé la résolution graphique de systèmes d'équations différentielles ordinaires, nos objectifs sont atteints.