Mathieu MARÉCHAL

# Simulation Numérique II

Nom de code : Dém***ez-vous !

---

# Compte-rendu des projets

---

Encadrants : C. Desjouy, Samuel Raetz

L2 SPI

2018 - 2019

# Introduction

# Table des matières

# Projet 1 La balle

L'objectif de ce projet est de modéliser la trajectoire d'une ou plusieurs balles en prenant en compte les rebonds. La consigne divise le travail en plusieurs étapes qui consistent à, tout d'abord, calculer et animer la trajectoire sans rebonds d'une balle, puis de prendre en compte les rebonds. Enfin, le programme nécessite d'être capable de calculer plusieurs trajectoires à la fois. J'ai donc décidé de procéder comme cela, en rajoutant des étapes : j'ai tout d'abord cherché à calculer une trajectoire simple et sans animation. Ensuite, j'ai créé les codes successifs dans des fichiers séparés, en rajoutant successivement : les animations, les rebonds, et enfin, plusieurs balles.

## 1.1 Calculer la trajectoire

Tout d'abord, les premières constantes dont va dépendre la trajectoire sont crées :
— l'angle initial, $\theta_0$ nommé `a_0`
— la vitesse initiale $v_0$, notée `v_0`
— plus tard, le temps de départ $t_0$ (`t_0`) sera utilisé
Ensuite, on implémente les équations déterminant la position du projectile en fonction des constantes précédentes :

$$\begin{cases} d_x = & v_0 t cos(\theta) + d_{0_x} \\ d_y = & -\frac{1}{2} g t^2 + v_0 t sin(\theta) + d_{0_y} \end{cases}$$

Il est maintenant possible de calculer la trajectoire mais la première question qui se pose est : quand doit-on arrêter de tracer la trajectoire ? En effet, si on trace la trajectoire sur un temps `t_balle` fixé, on observera la trajectoire avoir une position $d_y$ négative, avec une pente très élevé, ce qui n'a pas du tout la forme d'une parabole. C'est le premier problème que j'ai rencontré. C'est donc quand le projectile touche le sol qu'il faut arrêter de calculer la trajectoire, quand sa position en y vaut $0 \Leftrightarrow d_y = 0$. Pour cela, il faut calculer le temps de course du projectile, en résolvant l'équation $d_y = 0$. Il est alors possible de désigner une variable `t_max` qui calcule cette solution, ce qui permet de définir la limite du vecteur de temps `t_balle`.

```
t_max = (2*v_0*np.sin(a_0))/g + t_0
t_balle = np.arange(t_0, t_max, 0.1)
```

Par la suite, on peut animer la trajectoire du projectile. Pour cela, c'est la `FuncAnimation` du module `matplotlib.animation` qui sera utilisée, car elle utilise une fonction pour animer les données calculées et non une boucle `for` comme utilise l'`ArtistAnimation`. Ce choix est justifié aussi par le fait que le programme sera mis sous la forme d'une classe par la suite.

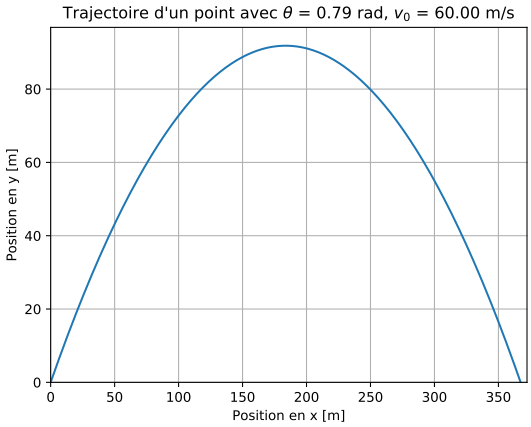
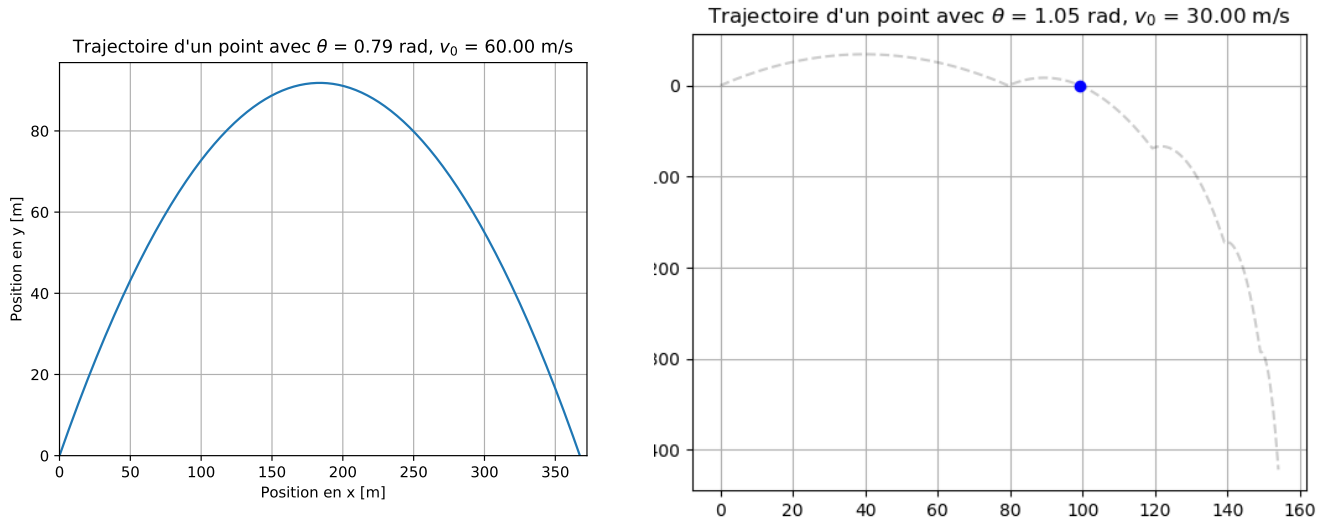



FIGURE 1 – Trajectoire fixe d'une balle




(a) A subfigure

(b) A subfigure

FIGURE 2 – A figure with two subfigures

### 1.2 Ajouter les rebonds

Implémenter les rebonds, paraît assez simple et c'est le cas ! Il suffit de déterminer une nouvelle vitesse initiale à la fin du calcul de la trajectoire et d'en calculer une nouvelle, ce qui est très aisé avec une boucle `for` itérant pour le nombre de rebonds souhaités.

Ainsi, on met sous cette boucle la totalité de la section du code réservé au calcul de la trajectoire développé précédemment. A la fin de la boucle, la vitesse initiale est multipliée par un coefficient de restitution $r$, `v_0 = r*v_0` caractérisant la rigidité du sol, et dans l'ensemble, l'élasticité du rebond, c'est-à-dire la quantité d'énergie de la balle absorbée par le sol. La vitesse de la balle va donc diminuer au fur et à mesure des rebonds successifs. L'autre donnée à conserver lors du calcul du rebond est la position horizontal de la trajectoire précédente. On stocke donc la dernière position en x telle que : `d_0_x = d_x` et on ajoute `d_0_x` en offset au calcul de la position.

Notre balle a donc obtenu la capacité de rebondir n fois, mais elle se sent toujours très seule.

### 1.3 Plus de balles

Dans cette partie, l'objectif est de pouvoir modéliser la trajectoire de **n** balles en simultané en définissant des constantes initiales aléatoires. Pour cela, il est possible de les réinitialiser avant le début en leur donnant une valeur aléatoire avec le module `random`.

Afin de simplifier le programme dans son ensemble et l'implémentation de cette fonctionnalité, le code sera mis sous la forme d'une classe.

## Projet 2 Le tube de Kundt

## Projet 3 L'interface du reste de ma vie

## Projet 4 Traitement du signal et imagerie

## Projet 5 Optique géométrique