

Introduction aux méthodes numériques et projet (PROJ0001):  
Modélisation de la trajectoire d'une balle de tennis

Premier bachelier en sciences de l'ingénieur  
Année académique 2021-2022. Travail réalisé par:

Loic Delbarre (S215072)      Rafik Lourmathi (S212097)  
Salman Guseynov (S216862)

Le 8 avril 2022

## Contents

<b>1</b>	<b>Question 1</b>	<b>3</b>
1.1	méthode sécante . . . . .	3
1.2	méthode de la bisection . . . . .	3
<b>2</b>	<b>question2</b>	<b>3</b>
<b>3</b>	<b>question3</b>	<b>4</b>
3.1	comparaison des deux methodes . . . . .	4
3.2	conception de trajectoirefilethorizontal . . . . .	4
<b>4</b>	<b>question 5</b>	<b>4</b>

# 1 Question 1

## 1.1 méthode sécante

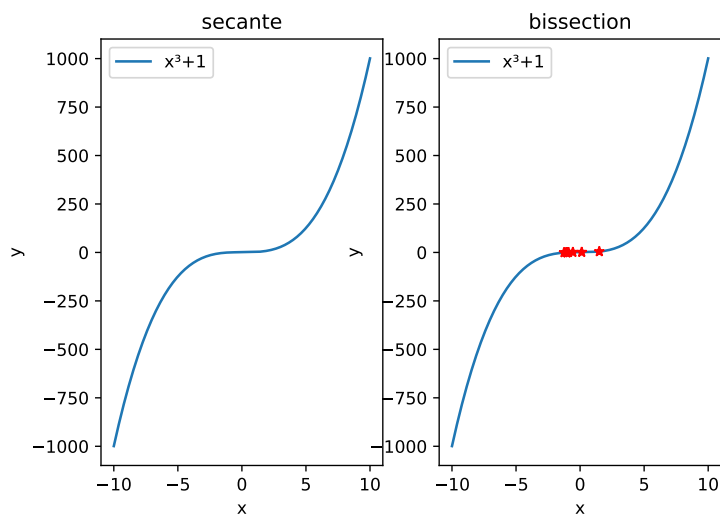
Nous avons donc implemente la methode de la secante et avons etudie les 3 cas qui peuvent se presenter a nous comme annonce dans l'annonce. Lorsque la secante converge et trouve un tuple contenant les coordonnees du zero approxime. Dans le deuxieme cas, afin de ne pas diviser par zero dans l'equation de la secante (ref), on retourne un status d'erreur. La methode de la secante ne nous permet pas d'etre assure de la convergence de la fonction, c'est pour cela que nous implementons un nombre d'iteration maximum comme gardien de boucle. Si jamais la fonction n'a pas converge apres le nombre max d'iteration, la fonction en deduit qu'il n'y a pas convergence. On a choisi le nombre d'iteration max comme etant de 100. On sait que la trajectoire d'une balle de tennis sera parabolique donc en faisant cette methode cela converge rapidement. Dans le cadre de ce cours, nous etudions des trajectoires paraboliques une vingtaine d'iteration devraient etre suffisantes mais au vu de la faible complexite nous avons mis 100.

Voici un exemple (ref) d'un graphique d'une fonction que l'on pourrait rencontrer lors de nos calculs de trajectoire. On remarque que avec 10 iteration on atteint assez rapidement une precision satisfaisante.

## 1.2 méthode de la bisection

La technique de la bisection est plus efficace que la secante car elle assure la convergence lorsque les bornes sont de signes contraires et definie partout entre ses bornes. Si la fonction n'est pas convergente on aura une sortie d'erreur.

`## il y a une erreur car la fonction ne converge pas`



# 2 question2

Afin de regrouper les differentes constantes utilisees dans le projet, nous les regroupons sous un fichier nomme const.py. On remarque egalement la possibilite d'une mise en evidence des

differentes constante permettant d eviter le calcul de multiplication de constante a chaque iteration(ref) On a profite de la caractéristique dtype des `numpy.array` pour choisir le nombre de decimales dans le calcul si besoin. Pour stocker les valeurs dans Euler, il a été plus judicieux de allouer statiquement un tableau de la taille du nombre d etape maximale et de le reduire lorsque l evenement (si la balle touche le sol) est produit. Dans le cas oppose lorsque l on realloait un tableau a chaque itteration avec `np.append` on a constate des pertes impresionantes de performance pour la reallocation de grands tableau. Pour un tableau de 800000 case, changer de technique a reduit le temps de moitie. D une autre part nous avons implemente l utilisation `solve_ivp` de `scipy`. au vu des valeurs par default et de leur precision nous avons decide de garder les valeurs. En optant pour la valeur de 0.001 pour `rtol` et la valeur 0.000001 pour `atol`, nous avons remarqué que avec ses valeurs par default aucun problème ne s est impose donc nous avons conserve ces parametres

$$\ddot{x} \cdot m = F_g + F_d + F_m$$

$$\ddot{x} = \rho \cdot \pi \cdot \frac{d^2}{8} \cdot \|\dot{x}\|^2 \left( \frac{C_m}{m} + \frac{C_d}{m} \right) - \frac{F_g}{m}$$

$$\ddot{x} = \frac{\rho \cdot \pi}{m} \cdot \frac{d^2}{8} \cdot \|\dot{x}\|^2 \left( \frac{1}{2 + 1.96 \frac{\dot{x}}{\|w\| \cdot d}} + C_d \right) - g$$

### 3 question3

#### 3.1 comparaison des deux methodes

En comparant `solve_ivp` et `euler` on se rends compte assez rapidement de la difference de temps. `solve_ivp` est instantane pour une precision de 10e-5 alors que `euler` prends déjà 6secondes. Et pour une precision 10 fois plus petite, `Euler` augmente drastiquement vers 60 alors que `Solve_ivp` reste instantane.

#### 3.2 conception de `trajectoirefilethorizontal`

On a decide de prendre un argument non obligatoire a la fonction `trajectoirefilethorizontal` pour gerer le cas ou on a un rebond de facon recursive. En effet, un rebond peut etre considere comme le point de depart d une nouvelle trajectoire. L avantage de cette technique, c est quelle permet d etre modulaire au niveau du rebond, si l on decide de ne plus prendre en compte le rebond, on ne devra pas changer de fonction.

Les eventment choisi pour l utilisation de `solve_ivp` on ete choisi de maniere analogique et non discrete car `solve_ivp` est capable d utiliser ces resultat pour converger plus rapidement et plus precisement. # question4 Pour impleter les diverses fonction, au vu de la tache repetitie, nous avons decide de separer la tache en plusieurs fonction (`get_cible`, `hauteur` et `getcible_rebond`) ...

### 4 question 5

On a pris en compte que la direction de la vitesse angulaire etait donnee en entree de la fonction. Nous decidons de calculer la hauteur pour chaque repartition avec une etape de 10e-2. A chaque etape on recalculera la hauteur. A la fin de toutes les repartitions, on essayera de retourner la repartition qui permet de trouver la valeur h la plus elevee. Pour l

etape , nous avons choisis  $10e-2$  .Avec  $10e-2$  on fait 100 fois des tours de boucle ce qui est la valeur minimale pour avoir une valeur en pourcentage

Pour savoir la repartition de l energie on se base sur l equation suivante (ref) et on en deduit les equation suivant (ref) premettant de calculer les diferentes fluctuation.

a titre de comparaison : Vu que la variation de vitesse angulaire est inexistente , on s est permi d hardcoder des zero dans le calcul `oderhs`

