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:

Le 15 avril 2023

Contents

1 Question 1

1.1 Méthode sécante

Nous avons donc implémenté la méthode de la sécante et avons étudié les 3 cas qui peuvent se présenter à nous comme annoncé dans l'énoncé.

Tout d'abord, envisageons le première possibilité : lorsque la sécante converge, elle renvoie un tuple contenant les coordonnées du zéro approximé.

Ensuite, dans le deuxième cas, afin de ne pas diviser par zéro dans l'équation de la sécante, on retourne un statut d'erreur.

La troisième plausibilité est que la méthode de la sécante ne nous permet pas d'être assuré de la convergence de la fonction. C'est pour cela que nous implémentons un nombre d'itérations maximum comme gardien de boucle. Si jamais la fonction n'a pas convergé après le nombre maximum d'itérations (`iteration_max`), la fonction en déduit qu'il n y a pas convergence et retourne un statut d'erreur.

On a défini le nombre maximal d'itérations égal à 100. On sait que la trajectoire d'une balle de tennis est parabolique. Donc, en utilisant la méthode de la sécante, on converge rapidement vers un zéro. Dans le cadre de ce cours, nous étudions des trajectoires paraboliques. Une vingtaine d'itérations devrait être suffisante. Mais, au vu de la faible complexité, nous avons fixé le nombre d'itérations à 100.Si la fonction était chronophage, cela aurait augmenté le temps de latence de récupération du résultat.Par exemple si la fonction prennait une seconde , avec 100 itérations on aurait obtenu un resultat après 100 secondes si jamais la fonction ne converge pas.

Voici un exemple(1) d'un graphique d'une fonction que l'on pourrait rencontrer lors de nos calculs de trajectoire. On remarque qu'avec 10 itérations, on atteint assez rapidement une précision satisfaisante.

1.2 méthode de la bissection

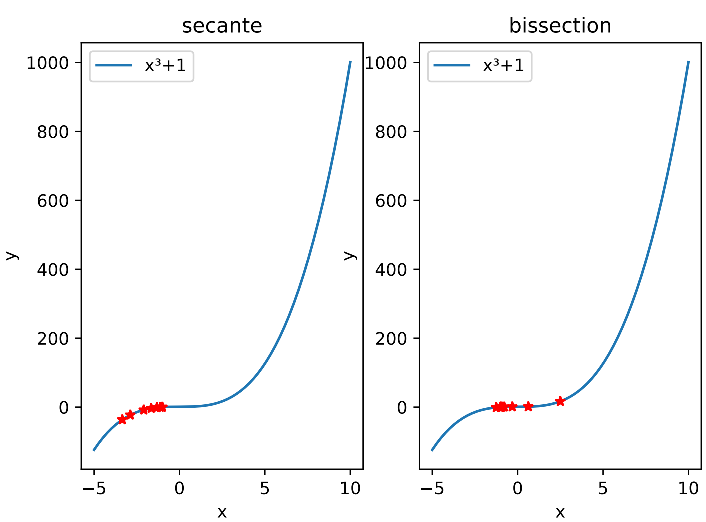

Figure 1: difference bissection et secante

Tout comme la sécante, 3 cas peuvent se présenter.

La premier cas est identique au précédant et nous retourne un tuple contenant les coordonnées d'une racine de la fonction.

La première hypothèse à respecter afin d'utiliser la méthode de la bissection est que les 2 valeurs (`x0` et `x1`) fournis en entrée à la fonction, possèdent des cordonnées y de signe opposé. Si cette condition n'est pas respectée, la fonction va parcourir le domaine à la recherche d'une valeur dont la fonction aura un signe opposé.Pour cela , elle va parcourir par dixieme le domaine compris entre les deux bornes.Si il n'y a toujours pas de valeur verifiant l'hypothese,la fonction retournera un statut d erreur.

La seconde hypothèse à respecter est celle stipulant que la fonction est défini partout entre ses valeurs d'entrée . Dans le cas où cette condition n'est pas respecté, on en déduit que la fonction ne converge pas on retourne alors un statut d'erreur.

2 Question 2

Afin de regrouper les différentes constantes utilisées dans le projet, nous les avons centralisées sous un fichier nommé `const.py` .

On a remarqué également la possibilité d'une mise en évidence des différentes constantes. Ainsi, on évite de recalculer le produit des constantes à chaque itération.

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

On voulait profiter de la caractéristique dtype des `numpy.array` pour choisir le nombre de décimales dans le calcul si besoin. Mais finalement, on s'est rendu compte que la valeur attribuée par défaut était amplement suffisante.

2.0.1 Euler

Pour stocker les valeurs dans Euler, il a été plus judicieux d'allouer statiquement un tableau de la taille du nombre d'étapes maximal et de le réduire lorsque l'événement (si la balle touche le sol) se produit. Dans la première option envisagée, on réallouait dynamiquement un tableau à chaque itteration avec `np.append`. Mais on a constaté des pertes impressionantes de performances pour la réallocation de grands tableaux. Pour un tableau de 800 000 cases , le fait de changer de technique a réduit le temps de moitié.

2.0.2 Solve_ivp

D'une autre part, nous avons implémenté l'utilisation `solve_ivp` de `scipy`. Au vu des valeurs par défaut et de leur précision, nous avons decidé de les conserver. Et donc les valeurs par défaut sont respectivement pour `rtol` de 0.001 et pour `atol` de 0.000001.

3 Question 3

3.1 Comparaison des deux méthodes

En comparant `solve_ivp` et Euler, on se rend compte assez rapidement de la différence de temps d'excution. `solve_ivp` est instantané pour une précision de 10e-5 alors qu'Euler prend déja 6 secondes. Et pour une précision 10 fois plus petite, Euler augmente drastiquement vers 60 alors que `solve_ivp` reste instantané.

3.2 conception de `trajectoirefilethorizontal`

On a décidé de prendre un argument non obligatoire à la fonction `trajectoirefilethorizontal` pour gérer le cas où on a un rebond de façon recursive. En effet, un rebond peut être

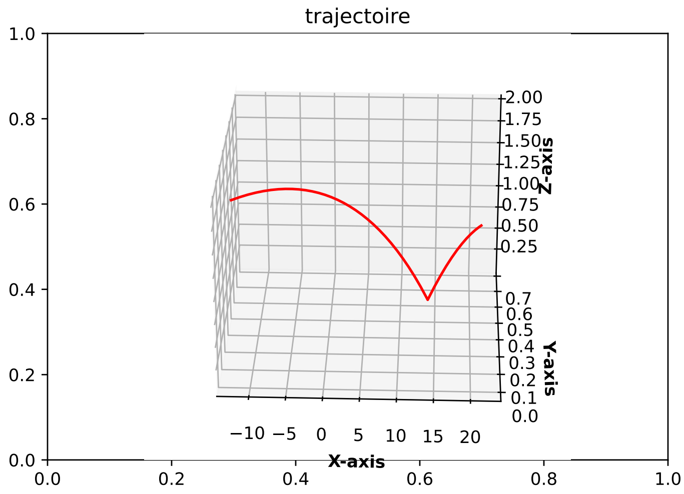

Figure 2: un exemple de trajectoire

considéré conmme le point de départ d'une nouvelle trajectoire. L'avantage de cette technique, c'est qu'elle permet d'être modulaire au niveau du rebond. Ainsi si on décide de ne plus prendre en compte le rebond, on ne devra pas changer de fonction. Les événements choisis pour l'utilisation de `solve_ivp` ont été déterminés de manière analogique et non discrète car `solve_ivp` est capable d'utiliser ces résultats pour converger plus rapidement et plus précisement.

Pour le choix des événements lors de la prise en compte du filet, nous avons opté pour `CubicSpline` comme dans le tutoriel avec `bc_type` en type natural afin d'obtenir un résultat plus naturel.

4 Question 4

4.1 Choix de la méthode

Le choix s'est porté sur la méthode de la bissection car celle-ci est sur de converger si on lui introduit 2 valeurs d'entrées qui respectent les hypothèses, contrairement à la sécante qui ne nous donne aucune information quant-à sa divergence.

4.2 Implémentation

Pour réaliser les diverses fonctionnalités, au vu de la tâche répétitive, nous avons décidé de séparer le travail en plusieurs fonctions : `Getciblehauteur`et `Getciblerebond`. La fonction `Getciblerebond` nous retournera la distance sur x de l'impact et `Getciblehauteur` nous retournera la hauteur à la ligne de fond.Nous nous sommes basée sur des `lambda` pour créer les fonctions que nous passeront en entrée à la `bissection` car nous les trouvions plus simple à comprendre que les `Inner function` de python. Pour faire également varier les différents éléments, nous avons implémenté `multinorm`,`multiomega`ainsi que `rotangle`.L'idée est de faire varier un paramètre d'entrée x pour y faire varier les diffentes variables sélectionnées. Pour les différentes fonctions, nous avons dû choisir des bornes différentes. Pour la rechercheHauteur, il etait spécifié dans les consignes de prendre une hauteur de 2 à 3 m. Pour la recherchevitesse, nous avons dû poser une valeur maximale à la norme, que nous avons fixée à 300m/s .Cette valeur nous parait suffiante au vu des conditions intiales et des valeurs moyennes de vitesse d'une balle de tennis.En effet,la vitesse d'une balle de tennis varie entre 30 et 60m/s,nous avons donc pris 300m/s car cela ne change rien et nous permet d avoir une plus grande bande d'action pour englober le plus de cas possible.La vitesse angulaire a été fournie dans l'énoncé. Pour l'angle demandé,nous avons pris de large valeurs car la bissection sera capable de determiner ses propres bornes.Ces valeurs oscillent logiquement entre 0 et $\frac{\pi}{2}$.En effet,si on ne tire pas dans cet intervalle,le tir ne serait que rebondir dans le même coté du filet.

5 Question 5

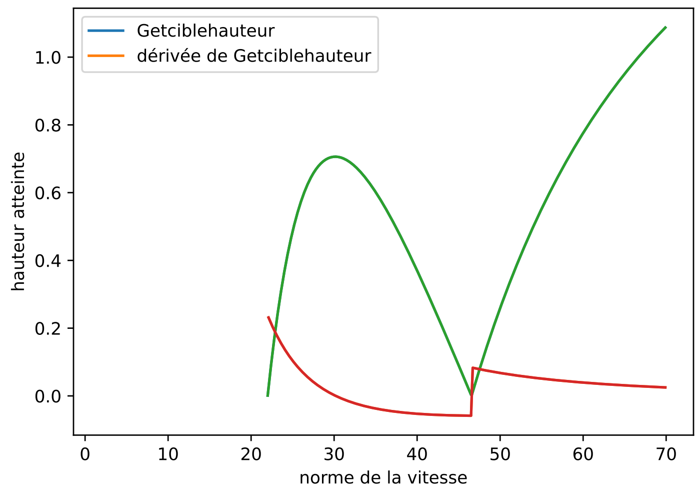

Figure 3: dérivée de la fonction 'Getciblehauteur'

Pour trouver le meilleur ratio de vitesse et de vitesse angulaire,nous nous sommes basés sur la dérivée de la fonction `Getciblehauteur`,en effet lorsque celle ci aura atteint sa valeur nulle,cela signifiera que nous sommes à la hauteur maximale pouvant etre atteinte.Il faut cependant négliger la plage de valeurs ne permettant pas d atteindre la ligne de fond.Pour cela , nous allons parcourir les valeurs possible jusqu'a ce que la fonction commence a fluctuer.C'est a partir de ce moment la que nous nous baserons sur le resultat obtenu d une bissection englobant la zone de fluctuation. Voici ici (3) un graphique de la hauteur finale en fonction de la vitesse. Ce graphique démontre clairement que la dérivée s'annule bien uniquement lorsque la balle est à l'apogée de la hauteur maximale.