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

1 Question 1

1.1 méthode sécante

Lors de l'implémentation de la sécante, 3 cas peuvent se présenter que nous allons analyser.

Le premier cas est celui que l'on désire, la secante converge et nous retourne un tuple contenant les coordonnées d'une racine réelle de la fonction.

Le cas suivant survient lorsque la fonction tente d'effectuer une opération impossible , en l'occurrence, on pourrait tomber sur cas ou l'on essaye de diviser par 0. La fonction retourne un message d'erreur suivi d'un tuple aux coordonnées aléatoire.

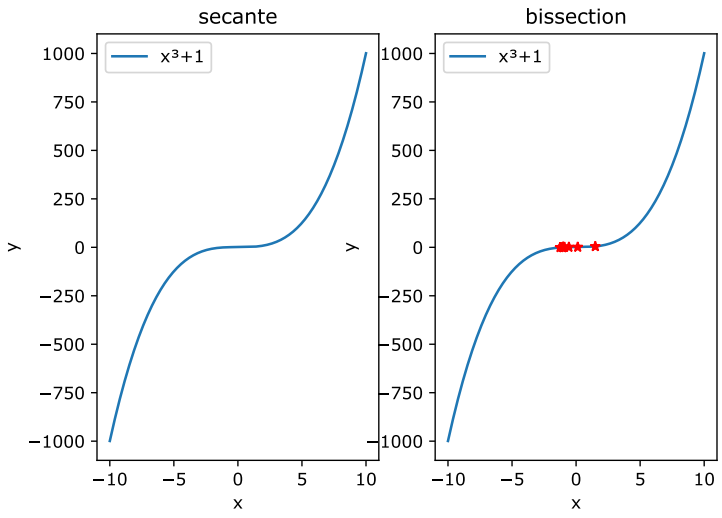
Le cas le plus problématique est celui ou la fonction converge infiniment lentement voire pas du tout. En effet, la méthode de la secante ne nous permet pas d'etre assuré de la convergence de la fonction. C'est pour cela que nous avons décidé d'attribuer un nombre maximum d'itération à notre boucle et conclu que si on ne trouve pas de racine après ce nombre d'itérations, la fonction ne converge pas. Nous avons décidé de fixer le nombre d'itérations maximum à 100. On se doute que la fonction représentant la trajectoire d'une balle de tennis sera plutot parabolique donc 100 itérations devraient largement suffire. Nous pourrions mettre un nombre d'iterations inférieure à 100 cependant la complexité en temps entre 30 , 50 ou 100 itérations est pratiquement identique.

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

1.2 méthode de la bissection

La technique de la bissection 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 aurra une sortie d erreur.

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



2 question2

Afin de regrouper les differentes constantes utilise dans le projet. , nous les regroupons sous un fichier nome const.py .On remarque egalement la possibilite d une mise en evidence des differentes constante permetant d eviter le calcul de multiplication de constante a chaque iteration(ref) On a profite de la carracteristique dtype des `numpy.array` pour choisir le nombre de decimales dans le calcul si besoin.Pour stoquer les valeur dans Euler , i l a ete 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 reallouait un tableau a chaque itteration avec `np.append` on a constate des pertes impresionantes de performance pour la realocation 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 defaut et de leur precision nous avons decide de garder les valeur En optant pour la valeur de 0.001 pour rtol et la valeur 0.000001 pour atol , nous avons remarqué que avec ses valeur par defaut aucun problème ne s est impose donc nous avons conserve ces parametres

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

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

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

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 déja 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 conmme 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.

Pour le choix des evenement,pour la prise en compte du filet , notre choix s est porté sur `CubicSpline` et comme `bc_type` nous avons choisit le type natural pour un resultat pour

4 question4

Pour impleter les diverses fonction , au vu de la tache repetitie , nous avons decider de separer la tache en plusieurs fonction (`get cible hauteur`et `getcible rebond`) ...

5 question 5

On a pris en compte que la direction de la vitesse angulaire etait donne en entree de la fonction. Nous decidons de calculer la hauteur pour chaque repartition avec une etape de 10e-2 . A chqaue etape on recalcuera 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 foi s 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`