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

Profs O. Brüls, Q. Louveaux, F. Nguyen

Février 2022

Introduction

Dans ce projet, nous allons simuler la trajectoire d'une balle de tennis lors d'un service (ou de tout autre coup). Pour ce faire, nous allons résoudre un système d'équations différentielles ordinaires. Nous donnerons les différentes vitesses et positions initiales de la balle et grâce à la résolution numérique du système d'équations différentielles, nous pourrons en déduire le reste de la trajectoire. Afin de calculer une trajectoire la plus proche possible de la réalité, nous tiendrons compte des différentes forces qui agissent sur la balle : la gravité, la force de frottement de l'air, et la force de portance générée par la rotation de la balle sur elle-même, cette force qui est générée par l'effet que l'on donne à la balle, connu sous le nom de slice ou lift selon le sens dans lequel la balle tourne.

Modèle

Le principe de base à appliquer est la loi de Newton : l'accélération d'un objet est égale à la somme des forces appliquées, divisée par la masse de l'objet. Rappelons que l'accélération est la dérivée de la vitesse, qui est elle-même la dérivée de la trajectoire.
On note le vecteur (tri-dimensionnel) de la position de la balle par x, ici

exprimé en m.
Le vecteur (tri-dimensionnel) de la vitesse (m/s) est $\frac{dx}{dt} = \dot{x}$.
Le vecteur (toujours tri-dimensionnel) de l'accélération (m/s^2) est $\frac{d\dot{x}}{dt} = \ddot{x}$.
La première chose à faire est de définir un repère. Vous êtes libre de choisir le repère que vous désirez. Pour la suite de cet enoncé, on définit le repère de la manière suivante :
1ère composante (x_1) : sur la longueur de terrain. On considère le point 0 au filet.
2ème composante (x_2) : la largeur de terrain. On considère le point 0 au centre.
3ème composante (x_3) : l'altitude. On considère le point 0 sur le sol.
Vous pouvez vous référer à la Figure 1 pour la définition des 2 premières composantes du repère.
A présent, il faut déterminer les forces qui s'appliquent sur la balle en mouvement. Dans notre cas, on considère que trois types de force s'appliquent sur la balle. Les valeurs des différentes constantes et les unités sont données en annexe.

1. La gravité
C'est la plus simple de toutes à modéliser. On a

$$F_g = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}.$$

2. Le frottement de l'air (ou traînée)
Cette force s'applique dans la direction opposée à la vitesse $\dot{x}$. On peut appliquer le modèle suivant pour la calculer. Son intensité est proportionnelle au carré **de la norme** de la vitesse, à la surface de contact à l'air et à la densité de l'air ρ. On a

$$\|F_d\| = C_d \rho \pi \frac{d^2}{8} \|\dot{x}\|^2.$$

Les valeurs des différentes constantes sont données en annexe. C_d est une constante, ρ est la densité de l'air et d le diamètre de la balle.

3. L'effet Magnus (dû à la rotation de la balle sur elle-même)
Cette force ne s'applique que quand la balle tourne sur elle-même. La différence de pression au sommet et en dessous de la balle induit une force, similaire

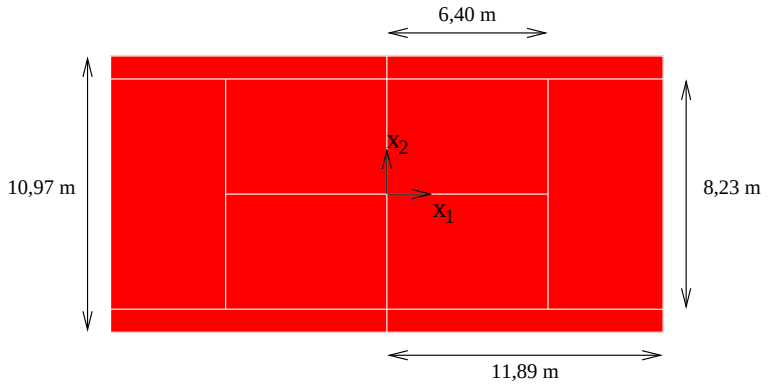

FIGURE 1 – Les dimensions du terrain de tennis et le repère horizontal

à la portance d'une aile d'avion. On modélise la rotation par un vecteur indiquant l'axe instantané de rotation et suivant la règle de la main droite : sur un plan horizontal, une balle qui tourne dans le sens des aiguilles d'une montre a un axe de rotation qui pointe vers le bas. La direction de la force est la même que le **produit vectoriel** de l'axe de rotation et du vecteur vitesse. L'intensité de la force induite par l'effet Magnus peut être modélisée de la manière suivante

$$\|F_m\| = C_m \rho \pi \frac{d^2}{8} \|\dot{x}\|^2.$$

Mais la valeur C_m dépend du rapport de la norme de la vitesse divisée par la vitesse de rotation de la balle. Des expériences réalisées sur des balles de tennis [1] ont montré qu'une bonne approximation de C_m est donnée par la formule

$$C_m = \frac{1}{2 + 1.96 \frac{v}{\|\omega\| d}}.$$

Dans cette expression, d est le diamètre de la balle, $v = \|\dot{x}\|$ est la norme de la vitesse et $\|\omega\|$ est la norme du vecteur de la vitesse angulaire. A nouveau, la valeur des constantes est donnée en annexe.

Résoudre un système d'équations différentielles de second ordre

Comme il s'agit d'un système d'équations différentielles de second ordre et que les méthodes vues au cours le sont pour des systèmes de premier ordre,

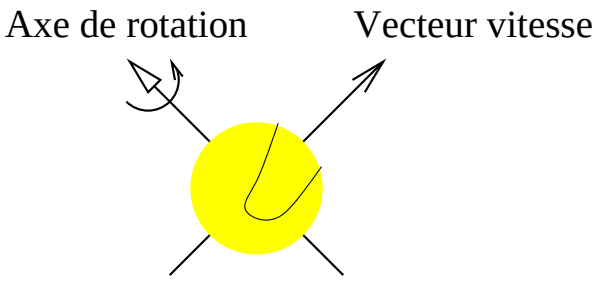

FIGURE 2 – La direction de l'effet Magnus en fonction de l'axe de rotation et du vecteur vitesse

il conviendra de créer des variables supplémentaires afin de transformer le problème en système de premier ordre. Si on veut résoudre

$$\ddot{x}(t) = f(x, \dot{x}, t),$$

on crée une nouvelle variable

$$y(t) = \begin{pmatrix} \dot{x}(t) \\ x(t) \end{pmatrix}$$

et on résout

$$\dot{y}(t) = \begin{pmatrix} f(x, \dot{x}, t) \\ \dot{x}(t) \end{pmatrix}.$$

Question 1

A la fin du projet, nous aurons besoin de résoudre une équation non-linéaire. La première question du projet vise à implémenter deux méthodes de recherche de racine qui ont été vues au cours, à savoir la méthode de la bissection et la méthode de la sécante. Ces implémentations doivent être le plus générique possible et donc pouvoir fonctionner pour tout type de fonction et avec le moins d'hypothèses de départ possibles. On demande d'implémenter

dans un module `RechercheRacine` les deux fonctions

```
x, statut = secante(f, x0, x1, tol)
```

et

```
x, statut = bissection(f, x0, x1, tol)
```

qui permettent de rechercher la racine d'une fonction Python `f(x)`, avec une tolérance définie par la valeur `tol`, à partir de deux valeurs initiales `x0` et `x1` selon la méthode de la sécante et de la bissection. Il est primordial de respecter le format demandé ci-dessus étant donné que ces fonctions seront testées automatiquement. En sortie, les deux fonctions doivent retourner un objet de type "list" avec deux valeurs. Dans le cas où la recherche a correctement convergé et donne une approximation de la racine à la tolérance près, la première valeur `x` contient la valeur de la racine et la deuxième valeur `statut` contient la valeur 0. Dans le cas où l'utilisateur a rentré deux valeurs qui ne satisfont pas les hypothèses de la méthode, il vous est demandé d'afficher un message d'erreur et d'affecter à la variable de sortie `statut` la valeur 1. Dans ce cas, `x` contient n'importe quelle valeur. Dans le cas où la méthode de la recherche de racine ne converge pas (par exemple parce que la fonction n'a pas de racine), vous devez afficher un message d'erreur et affecter la valeur -1 à la variable `statut`. Dans ce cas, à nouveau, n'importe quelle valeur peut être affectée à la variable de sortie `x`.
Dans votre implémentation, prenez soin (i) de vérifier les hypothèses de chacune de ces méthodes et de prévoir un comportement adéquat du code dans le cas où ces hypothèses ne sont pas rencontrées, de manière à fournir une solution dans un maximum de cas, (ii) de veiller à réduire autant que possible le nombre d'appels de la fonction `f(x)` à l'intérieur des fonctions `secante` et `bissection`, et ce pour des raisons d'efficacité. L'utilisation du `profiler` de `Spyder` vous permettra d'analyser l'efficacité de votre code.

Question 2

Dans cette question, nous allons implémenter la résolution du système d'équations différentielles ordinaires qui modélise la trajectoire de la balle. La première étape est d'implémenter une fonction qui, en calculant les différentes forces qui agissent sur la balle, renvoie les différentes dérivées des vecteurs position et vitesse.

Créer une fonction `oderhs` qui reçoit en entrée le temps, les trois composantes de la position de la balle, les trois composantes de la vitesse de la balle, ainsi que les trois composantes de l'axe de rotation de la balle. Cette fonction renvoie en sortie les dérivées de chaque entrée : les dérivées de la position qui sont simplement données par les vitesses respectives, les dérivées des vitesses qui sont données par la résultante des trois forces mentionnées plus haut divisées par la masse de la balle. On supposera que la vitesse de rotation de la balle est constante. Dès lors, les dérivées de l'axe de rotation sont nulles.
Plus explicitement, on demande d'implémenter une fonction `oderhs` qui prend en entrée le temps t et un vecteur de neuf composantes représentant dans l'ordre les différentes valeurs courantes des variables $x(t)$, $\dot{x}(t)$ et $\omega(t)$ et qui donne en sortie un vecteur de neuf composantes représentant les dérivées :

```
dy = oderhs(t, y)
```

où `y` $=[x, \dot{x}, \omega]$ et `dy` $=[dx/dt, d\dot{x}/dt, d\omega/dt]$.
Utiliser la fonction `oderhs` pour résoudre le système d'équations différentielles de deux manières :

- En implémentant la méthode d'Euler explicite,
- En utilisant la fonction `solve_ivp` de Python (librairie `SciPy`).

Dessiner le graphique de la trajectoire en fonction du temps sur une période d'une seconde en utilisant une position initiale de $x(0) = (-11.89, 0, 2)$, $\dot{x}(0) = (50, 1, 0)$, $\omega(0) = (30, 15, 0)$. Dans un premier temps, vous pouvez négliger le filet et la hauteur du sol et représenter la trajectoire suivie hypothétiquement pendant une seconde.

Question 3

Dans cette question, nous allons maintenant prendre en compte les contraintes physiques de l'environnement : le filet, le sol et donc le rebond. Vous devez d'abord vérifier si la balle passe au-dessus du filet. Dans un premier temps, le filet sera considéré comme horizontal de hauteur 1 mètre. Dans un second temps, vous devez considérer le filet comme arrondi. Déterminez une courbe raisonnable du filet à partir du règlement stipulant que le filet doit avoir une hauteur de 1,07 m à une distance de 0,914 m à l'extérieur des lignes de simple et une hauteur de 0,914 m au centre du terrain. Dans le calcul de la trajectoire, vous devez donc maintenant déterminer si la balle passe au-dessus du filet. Si tel n'est pas le cas, la trajectoire est interrompue au

niveau du filet. Si tel est le cas, vous devez trouver le point de rebond et calculer la deuxième partie de la trajectoire jusqu'à l'autre bout du terrain. Lors du rebond, seul le vecteur vitesse est modifié. La composante verticale de la vitesse v_z change de signe et est multipliée par un coefficient de restitution $e \in [0, 1]$ qui représente l'énergie dissipée au rebond. On considère que les composantes horizontales (v_x et v_y) restent inchangées. Enfin, on souhaite avoir un graphique de la trajectoire que la balle a suivie avec les valeurs initiales entrées.

Pour l'évaluation continue, vous devez écrire une fonction qui calcule la trajectoire de la balle, suppose un filet horizontal à hauteur de 1 mètre , et renvoie la position de la balle après T secondes. Cette fonction sera implémentée dans le module `Trajectoire` et suivra les spécifications

```
position = trajectoireFiletHorizontal(yInit,T)
```

où `yInit` est le vecteur initial à 9 composantes et `T` est le temps où la fonction doit renvoyer la position. Si la balle a été arrêtée par le filet dans l'intervalle de temps considéré, la fonction renverra la position $(0, 0, 0)$.

Question 4

Dans cette question, nous allons appliquer les méthodes numériques de la question 1 afin de trouver un coup atteignant une cible donnée.
On vous demande d'écrire des fonctions qui prennent en entrée un ensemble de conditions initiales $y0 = (x(0), \dot{x}(0), \omega(0))$, et qui calculent en sortie un paramètre permettant d'arriver à une cible donnée dans le camp adverse. Par exemple, on peut vouloir faire un coup qui touche la ligne de service ou faire un coup qui rebondit sur la ligne du fond (`cibleRebond`), ou faire un coup qui rebondit et atteint une hauteur de un mètre au niveau de la ligne du fond (`cibleHauteur`). Chaque fonction fera varier un paramètre différent : l'angle initial en radians de la vitesse la balle dans le plan vertical de la vitesse (et où la norme de la vitesse est conservée par rapport à $\|\dot{x}(0)\|$), la norme de la vitesse initiale de la balle (mais où la direction est conservée par rapport à $\dot{x}(0)$), la hauteur de frappe, la norme de la vitesse angulaire de rotation de la balle (où la direction de l'axe de rotation $\omega(0)$ est conservée).

```
angle, statut = rechercheAngle(y0,cibleRebond)
angle, statut = rechercheAngle2(y0,cibleHauteur)
normeVitesse, statut = rechercheVitesse(y0,cibleRebond)
normeVitesse, statut = rechercheVitesse2(y0,cibleHauteur)
hauteur, statut = rechercheHauteur(y0,cibleRebond)
hauteur, statut = rechercheHauteur2(y0,cibleHauteur)
omega, statut = rechercheOmega(y0,cibleRebond)
omega, statut = rechercheOmega2(y0,cibleHauteur)
```

Comme à la question 1, la variable `statut` vaut 0 quand la fonction a réussi à trouver la valeur et vaut -1 ou 1 quand la fonction a rencontré une erreur ou n'a pas pu converger. Cette fonction sera également implémentée dans le module `RechercheRacine`. Ces fonctions doivent utiliser une des fonctions de la question 1.

Question 5

Dans cette question nous allons essayer de calculer la combinaison qui donne le meilleur coup. Nous fixons la position de frappe initiale à $x(0) = (-11.89, 0, 2)$. L'énergie cinétique de la balle est calculée comme $\frac{1}{2}mv^2 + \frac{1}{8}md^2\omega^2$. L'énergie cinétique initiale totale de la balle est fixée à une valeur E. La direction de frappe est également fixée à un vecteur à trois dimensions `direction`.
Il vous est demandé d'écrire une méthode numérique qui trouve la répartition de l'énergie cinétique entre vitesse de translation et vitesse de rotation qui donne lieu à une trajectoire de balle la plus haute possible après le rebond au niveau de la ligne de fond.

Annexe : valeur des constantes

Symbole	Nom	Valeur
d	Diamètre de la balle	0,065 m
m	Masse de la balle	0,058 kg
ρ	Masse volumique de l'air	1,2 kg/m^3
C_d	Coefficient de traînée	0,65
C_m	Coefficient de Magnus	$\frac{1}{2+1.96\frac{v}{\omega d}}$
ω	Vitesse de rotation	$[0, 300]$ rad/sec
h_0	Hauteur de frappe	$[2m, 3m]$
h_f	Hauteur du filet	≈ 1 m
g	Gravitation	9,81 m/s^2
e	Coefficient de restitution	$[0.5, 0.8]$

Consignes

- Le travail comporte un code de calcul Python et un rapport d'une longueur de 10 pages maximum.
- Le code doit être correct et écrit par vous (ce que nous vérifierons à la présentation orale).
- Le code doit être soigné et commenté.
- Le code doit utiliser au maximum les possibilités vectorielles de numpy.
- Pour toute fonction, nous sommes susceptibles de vous demander de montrer un *profile* Python et de l'interpréter.

Critères d'évaluation

La note finale individuelle sera définie sur base de l'examen oral en prenant en compte la note de l'évaluation continue (milestones) ainsi que la note du rapport et du code Python. Un des objectifs principaux de ce cours étant l'apprentissage de la programmation des méthodes numériques en Python, il est indispensable que chaque étudiant et chaque étudiante réussisse l'examen oral, qui portera notamment sur la maîtrise de Python et de votre code, afin de réussir globalement.

Evaluation continue

Deux "milestones" permettront de vérifier votre état d'avancement en cours de projet.

— Vendredi 4 mars : vérification de la question 1 (fonctions `secante` et `bissection`).
— Vendredi 18 mars : vérification partielle de la question 3 en particulier la fonction `trajectoireFiletHorizontal`.

Ces fonctions seront testées automatiquement. Pour cette raison, il est primordial de respecter les consignes et le format demandé en termes de nom de fonction, de nom de module, de variables d'entrée et de sortie. Un non-respect des consignes sera sanctionné par une note de 0/10.

Rapport et code Python

Un fichier .zip par groupe comprenant un rapport au format PDF et accompagné des fichiers .py de votre programme doit être soumis via la plateforme de soumission au plus tard pour le vendredi 8 avril à 23h59. Le nom du fichier .zip et le nom du fichier .pdf doivent respecter le format suivant : "Numero-Groupe_NomA_NomB_NomC.xxx" (exemple : l'archive "27_Dupond_Beckers_Bastin.zip" doit inclure le fichier "27_Dupond_Beckers_Bastin.pdf").

— La longueur du rapport ne peut dépasser 10 pages et ne doit pas comporter d'introduction.
— Pour chaque question, les résultats obtenus doivent être illustrés.
— La justification des choix numériques est très importante. Pensez à expliquer les choix qui vous ont semblé cruciaux.
— La forme du rapport est prise en compte. Il est recommandé de suivre les règles de bonne pratique pour la réalisation d'un rapport scientifique qui sont présentées dans le podcast correspondant. Le nombre de pages étant limité, il est inutile de répéter l'énoncé. Allez donc à l'essentiel.
— La qualité du code (efficacité et soin) est également considérée dans l'évaluation.

Examen oral

L'examen oral est **individuel** et dure 10 minutes. Vous devez faire une démonstration du programme de votre groupe et répondre à des questions supplémentaires. Les éléments suivants seront pris en considération :

— la maîtrise de Python en tirant au sort dans une liste de questions disponibles avant l'examen ;

— la maîtrise du programme réalisé par votre groupe en tirant au sort dans une liste de questions disponibles avant l'examen ;
— les justifications et éclaircissements par rapport aux choix réalisés et aux résultats présentés dans le rapport et dans le code ;
— la maîtrise des notions théoriques vues au cours en lien avec votre travail et la maîtrise générale du projet.

Deuxième session

Les groupes qui, en première session, ont obtenu pour le rapport et le code une note suffisante sont, s'ils le souhaitent, dispensés de remettre un nouveau code et un nouveau rapport lors de la session de septembre. Dans ce cas, seul l'oral doit être représenté et compte pour 100% de la note finale. Pour les autres groupes, un nouveau code et un nouveau rapport doivent être remis 5 jours avant l'examen oral.

Références

[1] A. Stepanek. *The aerodynamics of tennis balls - The topspin lob.* Am. Journal of Physics **56**(2), 1988.