UN PETIT JEU DE BALISTIQUE

L'objectif de ce TD est de réaliser un petit jeu de balistique en n'utilisant que les outils vus jusqu'à présent.

Il n'est donc pas question d'introduire de nouvelles bibliothèques, même si elles seraient plus adaptées.

Règles du jeu :

On dispose d'un canon placé à l'origine : $(0,0)$ qui doit tirer sur une cible.

On peut choisir la vitesse initiale du boulet, ainsi que l'angle par rapport à l'horizontale.

La cible est placée aléatoirement et le but est de l'atteindre avec le minimum de coups possibles.

Remarque : Penser à tester les algorithmes au fur et à mesure de leur écriture.

1 PRÉAMBULES

Chargement de bibliothèques :

```
import pylab
from random import random    # donne un nombre aléatoire dans [0,1[
ion()                        # pour un tracé interactif
```

Variables :

- `v0` désigne la vitesse initiale du boulet choisie par le joueur.
- `vmax`$=20$ désigne la vitesse initiale maximale possible pour le boulet.
- `angle` désigne l'angle de tir *en degrés* choisi par le joueur.
- `g`$=9,8$ est l'accélération de la pesenteur.

On suppose que le boulet est soumis à la seule force de son poids, ainsi, il vérifie les équations :

$$\begin{cases} z'' &= -gz \\ x'' &= 0 \end{cases}$$

avec z son altitude et x son abscisse.

2 VERSION DE BASE

1) Tracé de la trajectoire

(a) À partir des conditions initiales, donner l'altitude z du boulet en fonction du temps.

(b) Déterminer l'instant de retour au sol du boulet.

(c) Calculer l'instant maximal de retour au sol pour les conditions les plus *défavorables*. On notera ce temps `tmax`.

(d) Créer une liste `temps` qui contient 200 valeurs uniformément réparties entre 0 et `tmax`.
On note `z` la liste des altitudes pour chacun des 200 instants et `x` la liste des abscisses. Construire ces deux listes.

(e) En déduire le tracé de la trajectoire du boulet.

2) Positionnement le la cible

(a) Définir une variable `cible` qui contient les coordonnées de la cible.
Elle doit être placée aléatoirement avec une abscisse entre 5 et 20 et une ordonnée entre $0,5$ et 5.

(b) Dans toute la suite, on fixera la taille de la fenêtre du graphique :

```
ylim([0,20]); xlim([0,20])
```

3) Tester si la trajectoire atteint la cible.

On ne cherche pas une résolution mathématique, mais une résolution approchée en s'aidant de trajectoire écrite sous forme de liste.

Pour cela

(a) Définir le rayon de la cible `rayon`$= 0{,}5$ (pour commencer).

(b) Définir l'indice i de la première abscisse après l'impact supposé dans la liste `x`.

(c) Vérifier si la distance entre l'ordonnée du boulet au temps i et celle de la cible est inférieure au rayon.

4) Mettre tout cela dans une seule fonction **jeu**, qui permet à l'utilisateur d'essayer jusqu'à ce qu'il atteigne la cible. La fonction renverra le nombre d'essais nécessaires.

Indications :

- Après chaque utilisation de `plot`, utiliser la fonction `pause(.01)` pour laisser à Python le temps de tracer le graphique.
- Proposer à chaque étape de rentrer une vitesse nulle pour arrêter le jeu. Dans ce cas, la fonction renvoie -1.

3 Ajout du vent

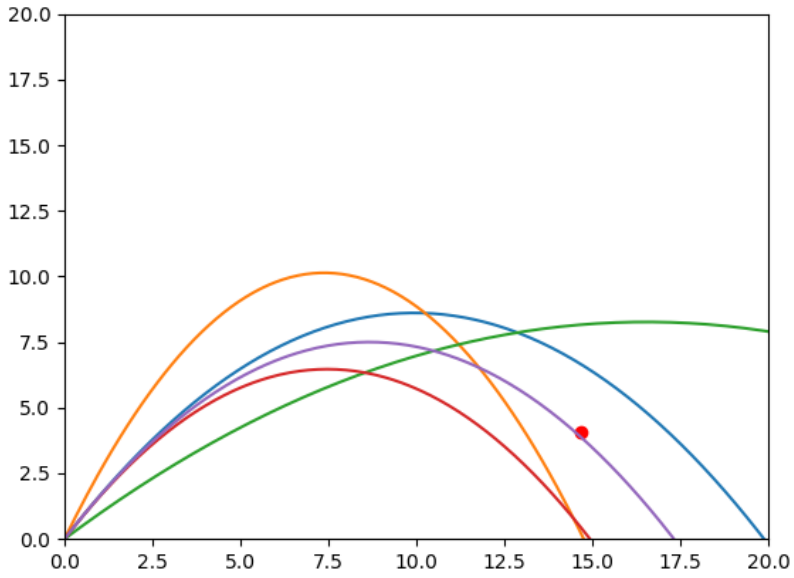
Proposer dans le jeu d'ajouter un vent constant. On pourra prendre un vent de vitesse aléatoire comprise entre 0 et `max/5`.

Indication : à chaque valeur de x dans la liste, retirer le déplacement cumulé du vent.

4 Améliorations

En s'aidant du cours en ligne, on peut demander un nom à l'utilisateur et enregistrer les scores dans un fichier.

Ensuite, il est possible de réaliser un tableau d'honneur.



BOUM !