But du TP : Modéliser un lancer de balle à l'aide d'un code python.

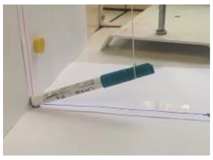
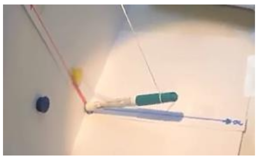
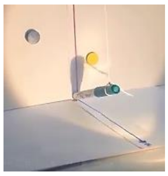
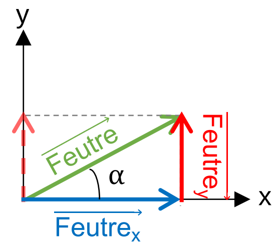
On veut étudier à l'aide d'une simulation Python, le mouvement du centre d'inertie d'une balle en <u>chute libre</u> (c'est-à-dire soumise <u>uniquement à son poids</u> $P = mg$ - on néglige les effets de l'air) afin de vérifier la seconde loi de Newton approximée.

Données : *champs de pesanteur terrestre* $g = 9{,}8\ m.s^{-2}$

<u>Projection d'un vecteur</u>

La projection de vecteur est un outil essentiel en physique, et particulièrement en mécanique, afin de déterminer les composantes *x* et *y* d'un vecteur.

Imaginons l'expérience suivante : On place l'extrémité d'un feutre, attaché à une ficelle à l'autre extrémité, sur le coin entre deux plans perpendiculaires de telle sorte à ce que le feutre fasse un angle avec les axes (image de gauche).
Si on éclaire le feutre directement au-dessus, une ombre se projette sur l'axe horizontal : c'est la projection orthogonale selon l'axe Ox (image du centre).
De même, si on éclaire le feutre directement en face, une ombre se projette sur l'axe vertical : c'est la projection orthogonale selon l'axe Oy (image de droite).

Le feutre et ses projections forment alors un triangle rectangle, dans lequel les règles de trigonométrie et le théorème de Pythagore s'appliquent :

<u>Trigonométrie</u> (CAH SOH TOA)

$$\cos\alpha = \frac{Adjacent}{Hypoténuse} = \frac{Feutre_x}{Feutre} \qquad \sin\alpha = \frac{Opposé}{Hypoténuse} = \frac{Feutre_y}{Feutre}$$

$$\tan\alpha = \frac{Opposé}{Adjacent} = \frac{Feutre_y}{Feutre_x}$$

<u>Pythagore</u>

$$Feutre^2 = Feutre_x^{\,2} + Feutre_y^{\,2}$$

<u>Document 1</u> : Décomposition des vecteurs position, vitesse et accélération **dans le cas d'une chute libre**

Comme tout vecteurs, les vecteurs position, vitesse et variation de vitesse peuvent être décomposé en leur différentes composantes, par exemple, sur l'horizontale (*x*) et la verticale (*y*).
Lorsque ces composantes sont exprimées en fonction du temps, on les nomme **équations horaires**.

<u>Position du centre d'inertie</u>

On considère un objet, représenté par un point M, en chute libre dans un champ de pesanteur uniforme $\vec{g}$, que lancé avec une vitesse initiale $\vec{v_0}$ faisant un angle α (alpha) avec l'horizontale.

Le vecteur position à l'instant t_i, $\overrightarrow{OM(t_i)}$, est repéré par rapport à l'origine O du système de coordonnées choisi pour l'étude du mouvement de telle sorte que ses composantes soient :

$$\overrightarrow{OM(t_i)} = \begin{cases} x(t_i) = v_0\cos(\alpha) \times t_i \\ y(t_i) = -\dfrac{1}{2}\,g\,t_i^2 + v_0\sin(\alpha) \times t_i \end{cases}$$

<u>Vecteur vitesse</u>

$$\overrightarrow{v(t_i)} = \begin{cases} v_x(t_i) = \dfrac{x(t_{i+1}) - x(t_{i-1})}{t_{i+1} - t_{i-1}} \\ v_y(t_i) = \dfrac{y(t_{i+1}) - y(t_{i-1})}{t_{i+1} - t_{i-1}} \end{cases}$$

<u>Vecteur accélération</u>

(= variation de vitesse par unité de temps)

$$\overrightarrow{a(t_i)} = \frac{\overrightarrow{\Delta v(t_i)}}{\Delta t} = \begin{cases} a_x(t_i) = \dfrac{v_x(t_{i+1}) - v_x(t_{i-1})}{t_{i+1} - t_{i-1}} \\ a_y(t_i) = \dfrac{v_y(t_{i+1}) - v_y(t_{i-1})}{t_{i+1} - t_{i-1}} \end{cases}$$

Une chronophotographie du mouvement étudié est donnée en annexe. L'échelle de distance est représentée par la flèche noire et l'échelle de vitesse par la flèche blanche.
La balle est lancée avec une vitesse initiale v_0 = 4,8 m.s^{-1} faisant un angle α = 68° avec l'horizontale.

1. En présentant correctement l'exercice de mécanique, faire le bilan des forces sur le système étudié et tracer les vecteurs force correspondant en rouge au point M_{17} de l'annexe, sans soucis d'échelle.

2. En utilisant la chronophotographie et en projetant la trajectoire sur l'axe, justifier que suivant la direction horizontale (Ox) le mouvement de la balle peut être considéré comme uniforme.

3. A l'aide de la même méthode, mais suivant la direction verticale (Oy), décrire l'évolution de la vitesse dans le mouvement de la balle.

Appeler le professeur pour évaluer les résultats

4. Tracer en bleu le vecteur vitesse initiale $\overrightarrow{v_0}$ au point M_0 en respectant l'échelle des vitesses indiquée sur l'annexe, puis vérifier en les mesurant que les projections de ce vecteur respectent bien :

$$\overrightarrow{v_0} = \begin{cases} v_x(0) = v_0 \cos(\alpha) \\ v_y(0) = v_0 \sin(\alpha) \end{cases}$$

Note : vérifiez bien ici que votre calculatrice est en degré, pas en radian : faites sin(90), si vous n'obtenez pas 1, vous êtes en radian !

5. A l'aide de l'échelle de distance indiquée sur l'annexe, <u>mesurer</u> les coordonnées (x_i ; y_i) des points M_3, M_5 et M_7 puis <u>calculer</u> les coordonnées des vecteurs vitesses aux points M_4 et M_6 et enfin les coordonnées du vecteur variation de vitesse au point M_5.
Tracer sur l'annexe en bleu les vecteurs vitesses $\overrightarrow{v_4}$ et $\overrightarrow{v_6}$ et en vert le vecteur variation de vitesse $\overrightarrow{\Delta v_5}$, en respectant l'échelle de vitesse.

Appeler le professeur pour évaluer les résultats

Le code python mvt_parabolique.py permet de modéliser la trajectoire de la balle, puis de tracer points et modèle sur un graphique.
Ouvrir le code et l'examiner. Une fiche méthode d'aide à python est fournis en fin de sujet.

6. Les lignes 31, 40 et 48 sont incomplètes (indiquées par ##A COMPLETER). Les compléter avec les expressions attendues pour les coordonnées sur l'axe (Oy) des positions, des vecteurs vitesse et des vecteurs accélération.
Attention, les expressions doivent être entrées de manière linéaire (comme dans une calculatrice collège, avec des parenthèses).
Aide : Vous pouvez vous inspirer des expressions données pour les coordonnées sur l'axe (Ox) pour la syntaxe.

Appeler le professeur pour vérifier le code
Après accord du professeur, exécuter le code.

7. Sur le graphique résultant de la simulation, identifier à quoi correspondent les flèches bleues et les flèches vertes. Justifier.

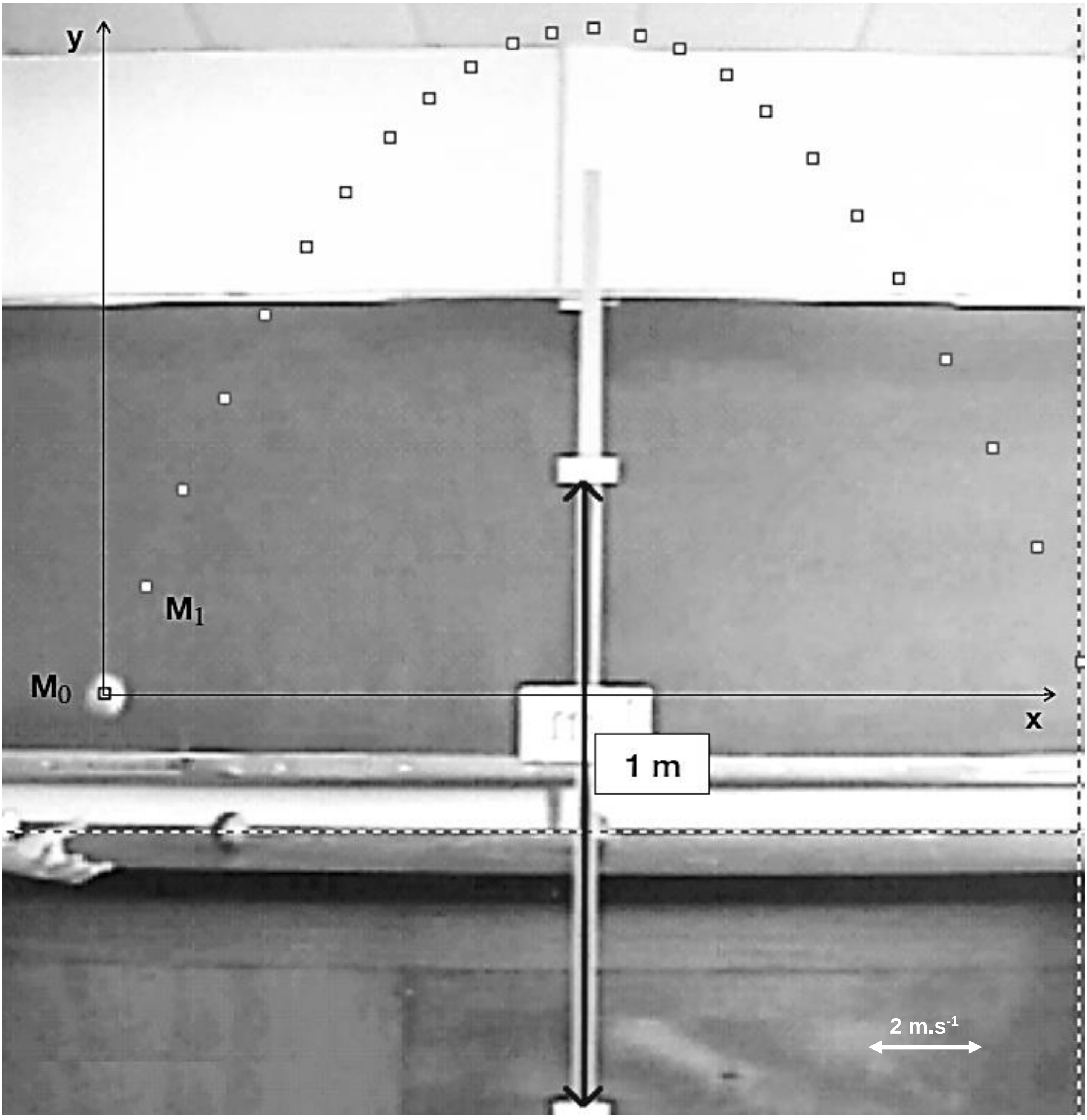
8. La simulation python affiche, dans la console, les valeurs de l'accélération à chaque instant. Ces valeurs sont-elles cohérentes avec celles trouvées en appliquant mathématiquement la seconde loi de Newton ? Justifier clairement en faisant le calcul.

Appeler le professeur pour évaluer les résultats

Annexe

Système : {balle}

Durée entre chaque image : τ = 0,04 secondes



Fiche Méthode	Quelques repères de programmation en Python

Noms : **Le nom des variables et des fonctions est sensible à la casse** (majuscules et minuscules sont traitées comme des lettres différentes) et ne doit pas contenir d'espace, de caractères spéciaux (sauf le sous-tiret '_') ou accentués.

Commentaires : A partir du caractère '#' et jusqu'à la fin de la ligne, tous les caractères sont ignorés par l'interpréteur Python (le logiciel qui exécute les instructions du programme). Ceci sert à mettre des informations destinées uniquement aux lecteurs humains.

Nombres à virgule : Les nombres à virgule sont représentés avec un point entre la partie entière et la partie décimale (ex : 2.5).

Les opérateurs de base :	Exemple
Opérations classique : + ; − ; * (la multiplication est notée *) ; /	`n = 2 * 3 + 5    # n vaut 11`
Mise à la puissance : **	`c = 4**3  # c vaut 4 à la puissance 3 donc 64`

Instruction	Exemple
Importer un module (pour pouvoir utiliser ses fonctions) `import nom_du_module` A mettre toujours en début de fichier	`import math  # fonctions usuelles de math` `import matplotlib.pyplot as plt`
Afficher une variable `print(variable)`	`h = 6.626e-34` `print(h)` Affiche 6.626e-34
Boucles	

Permet de répéter N fois un bloc de code
`for i in range(N) :`
→ `bloc` C'est le retrait (*l'indentation*) des lignes qui permet de définir le bloc à répéter
	`for i in range(N) :`
	`    a = a + i`
	`    print(i**3)`
	i va prendre toutes les valeurs entières de 0 à N-1

Listes	

Les listes numérotées (type `list`) :
Une liste est une structure permettant de stocker plusieurs valeurs de tous types en les rangeant selon leur numéro
	`t = []   # crée une liste vide`
	`t = [42, 'abc']  # crée une liste de deux valeurs (le nombre 42 et la chaîne 'abc')`
	`t = [0]*25  # crée une liste de 25 éléments ne contenant que des zéros`

Ajouter un élément à la fin d'une liste : `append()`	`t.append(b) # ajout l'élément b à la fin de la liste t`
Déterminer la taille de la liste : `len()`	`t = [4, 'kilo', 9, -1, 0]` `longueurListe = len(t)` longueurListe vaut 5
Accéder à un élément de la liste par son numéro `t[i]` i+1ème élément de la liste t (les indices vont de 0 pour le 1er terme à la taille de la liste - 1)	`t = ['x', 3.5, 42, 'Ampère']` `print(t[2])` Affiche 42 `t[0] = 7` t vaut maintenant [7, 3.5, 42, 'Ampère']

Lire ou utiliser les éléments d'une liste	`#pour tous les éléments de la liste V`
	`for i in range(len(V)) :`
	`    #calculer masse/V et ajouter à la liste rho`
	`    rho.append(masse/V[i])`