Terminale Spécialité φχ	Mouvement et interactions	**TP N°4 : Galilée et les plans inclinés**

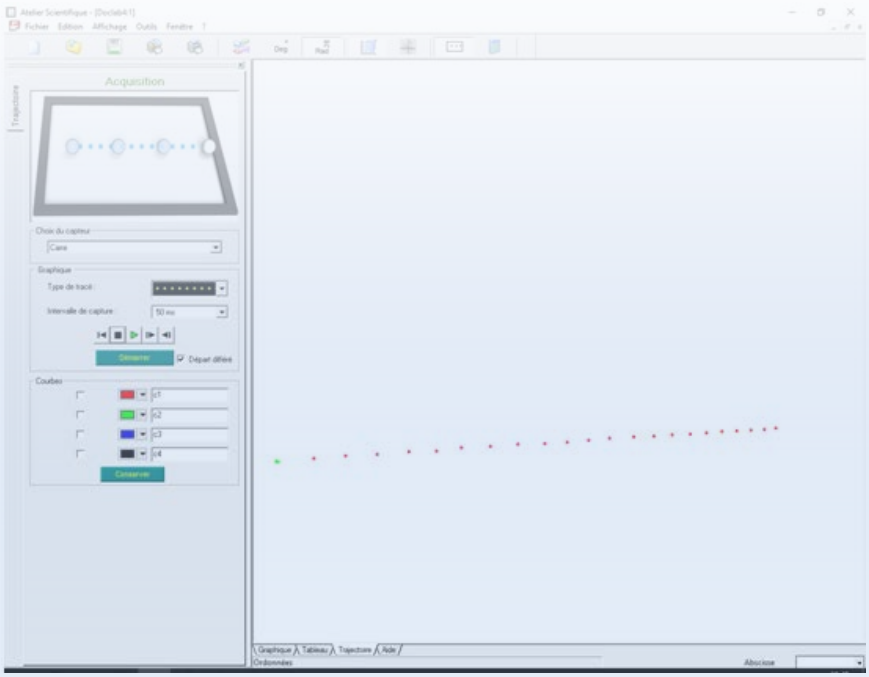
Objectifs : Réaliser une chronophotographie pour déterminer les coordonnées du vecteur position en fonction du temps et en déduire les coordonnées approchées ou les représentations des vecteurs vitesse et accélération.
Représenter, à l'aide d'un langage de programmation, des vecteurs accélération d'un point lors d'un mouvement.

Document 1 : Galilée l'expérience des plans inclinés

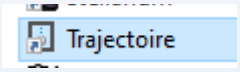
Vidéo Galilée l'expérience curiosphere tv : *https://www.youtube.com/watch?v=PZiTHgPxD88*
Florence, 1609. Dans une petite chambre, un homme s'affaire. Ce n'est pas un enfant, et pourtant il joue à un jeu étrange: il fait rouler des billes sur un plan incliné en faisant tinter des clochettes! Cet homme, c'est Galilée. Il cherche la loi de la chute des corps: pour tenter de percer ce mystère, il observe une pierre qui tombe. Que remarque-t-il ? Dans sa chute, elle accélère. Oui, mais de quelle manière ?

Document 2 : Matériel utilisé pour enregistrer des trajectoires

A l'aide d'un Inclinomètre numérique, régler une pente de ….%



- Dans le dossier PHYSIQUE & CHIMIE Lancer



- Choisir un intervalle de 50 ms
- Choir départ différé
- Lâcher la bille juste avant le début de l'enregistrement.
- Appuyer sur Echap
- Choir la bonne courbe et conserver
- Dans l'onglet tableau, relever les coordonnées t,x,y

Problématique :

Proposer un protocole pour enregistrer le mouvement d'une bille sur un plan incliné à 15° et déterminer les caractéristiques des vecteurs accélération.

APPEL n°1		
	**Appeler le professeur pour faire vérifier le protocole**	

<u>Document 3</u> : Script python à compléter pour tracer les vecteurs vitesse et accélération

A rendre sous Capytale : https://capytale2.ac-paris.fr/web/c/f67f-736647
(Code f67f-736647)

```
import matplotlib.pyplot as plt

t = [0.589, 0.639, ...]  # compléter avec la liste des dates du point 0 au point 9
X = [0.279, ...]  # compléter avec la liste des abscisses du point M
Y = [0.17, ...]  # compléter avec la liste des ordonnées du point M
nombre_mesures = len(t)


# Tracé de la trajectoire
plt.figure(1)  # Déclaration du graphique n°1
plt.scatter(X, Y, color="blue", marker='+')  # Tracé en nuage de points
plt.xlabel("x en m")  # Étiquette de l'axe des abscisses
plt.ylabel("y en m")  # Étiquette de l'axe des ordonnées
plt.grid()  # Affichage de la grille
plt.axis("equal")  # Pour avoir un repère orthonormé

# Tracé des vecteurs vitesse
echelleV = 1
Vx = []
Vy = []
for i in range(1, nombre_mesures - 1):
    delta_t = t[i + 1] - t[i - 1]  # formule pour déterminer l'intervalle de temps
    vx = (X[i + 1] - X[i - 1]) / (delta_t)  # formule pour déterminer les coordonnées vx

    vy = (...)  # formule à compléter pour déterminer les coordonnées vy de la vitesse

    print("Au point", i, ":(vx=", round(vx, 3), "; vy=", round(vy, 3), ")")
    Vx.append(vx), Vy.append(vy)  # ajoute les coordonnées du vecteur vitesse à la liste

# Représentation des vecteurs vitesse
    plt.quiver(X[i], Y[i], vx * echelleV, vy * echelleV, color="green")

# Tracé des vecteurs accélération

nombre_mesures = len(Vx)  # nombre d'intervalles pour les calculs d'accélération
echellea = 1

for i in range(1, nombre_mesures - 1):
    delta_t = t[i + 2] - t[i]  # formule pour déterminer l'intervalle de temps

    ax = (Vx[i + 1] - …..)# formule à compléter pour déterminer les coordonnées ax
    ay = ….  # formule à compléter pour déterminer les coordonnées ay de l'accélération

    print("Au point", i, ":(ax=", round(ax, 3), "; ay=", round(ay, 3), ")")
    Représentation des vecteurs vitesse
    plt.quiver(X[i+1], Y[i+1], ax * echellea, ay * echellea, color="red",)


plt.show()  # Affichage du graphique
```

Prolongement :

Q1. D'après Galilée, la vitesse de chute dépend elle de la masse de la bille ?

Q2. Proposer un protocole pour le vérifier.

Matériel utilisé :

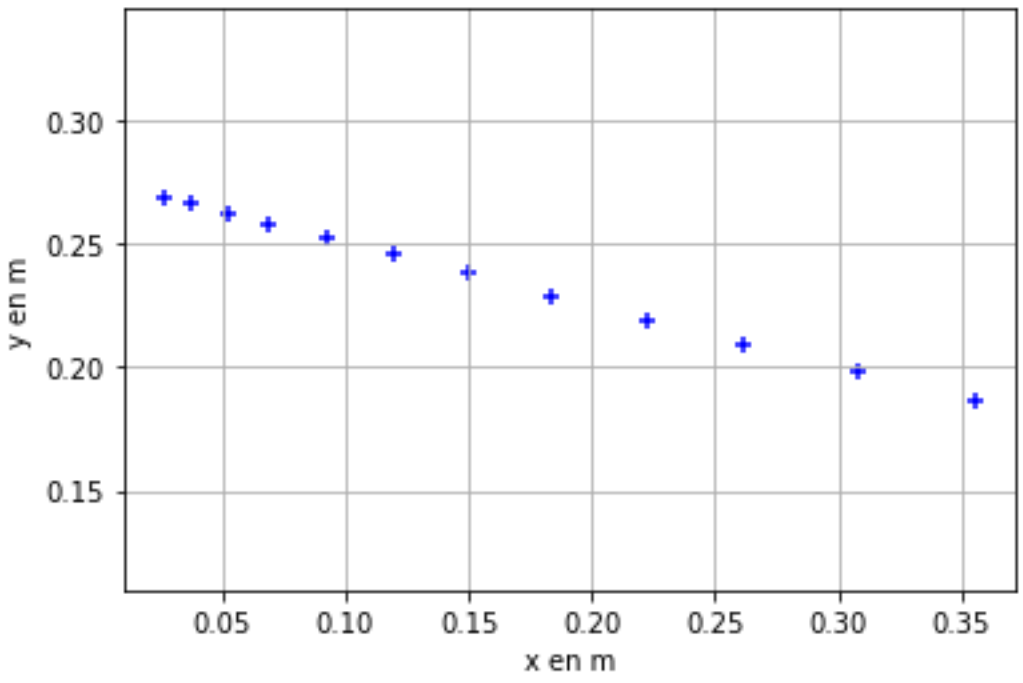
- **Détecteur infrarouge – pack énergie et mouvements**
- **Table à mobiles autoporteurs**
- **Appareil photo et logiciel de pointage**
- **Ordinateur sous W10.**
- **Le logiciel fourni : « Trajectoire »**
- **Plusieurs billes**
- **Une balance**

Activité Capytale : e31a-735310

Bille en verre m = 56 g diamètre 3,5 cm
Pente=15°
t=[0,0.049,0.1,0.145,0.191,0.238,0.284,0.33,0.376,0.422,0.469,0.517]
X=[0.026,0.037,0.052,0.069,0.092,0.12,0.15,0.183,0.222,0.262,0.308,0.356]
Y=[0.269,0.267,0.262,0.258,0.253,0.246,0.238,0.229,0.219,0.209,0.198,0.186]

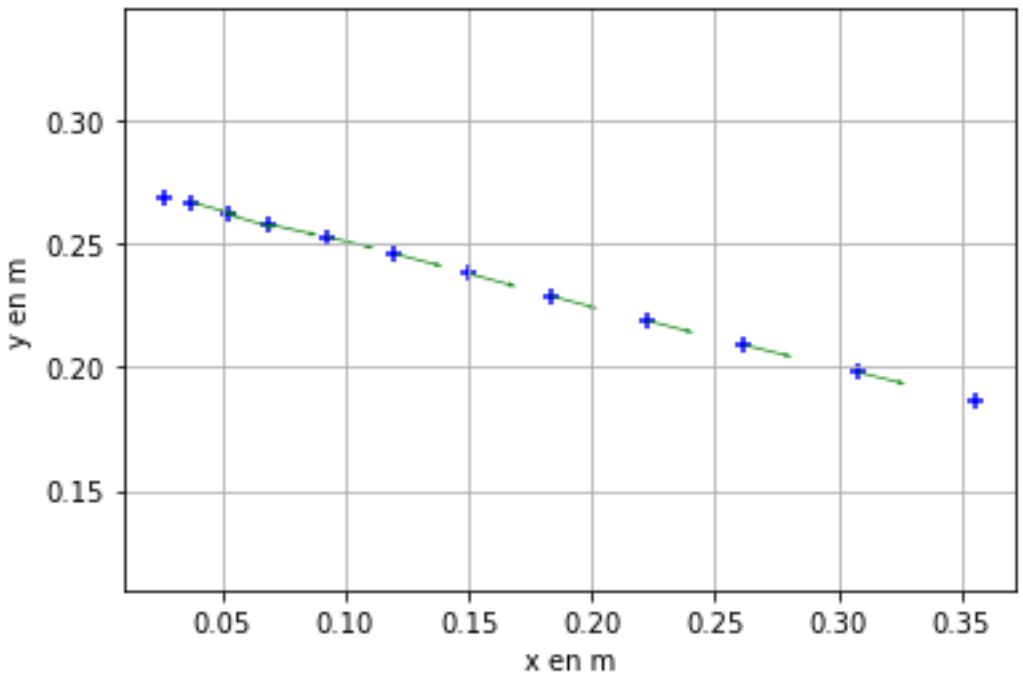
Trajectoire :

```
import matplotlib.pyplot as plt
t=[0,0.049,0.1,0.145,0.191,0.238,0.284,0.33,0.376,0.422,0.469,0.517]
X=[0.026,0.037,0.052,0.069,0.092,0.12,0.15,0.183,0.222,0.262,0.308,0.356]
Y=[0.269,0.267,0.262,0.258,0.253,0.246,0.238,0.229,0.219,0.209,0.198,0.186] # compléter avec la liste Y des ordonnées y du point M

nombre_mesures = len(t) # nombre d'intervalles pour les calculs de vitesse

# Tracé de la trajectoire
plt.figure(1)  # Déclaration du graphique n°1, indispensable pour tracer plusieurs représentations graphiques
plt.scatter(X, Y, color = "blue", marker = '+')  # Tracé en nuage de points avec indications minimales
plt.xlabel("x en m")  # Étiquette de l'axe des abscisses
plt.ylabel("y en m")  # Étiquette de l'axe des ordonnées
plt.grid()  # Affichage de la grille
plt.axis("equal")  # Pour avoir un repère orthonormé

plt.show()  # Affichage du graphique
```



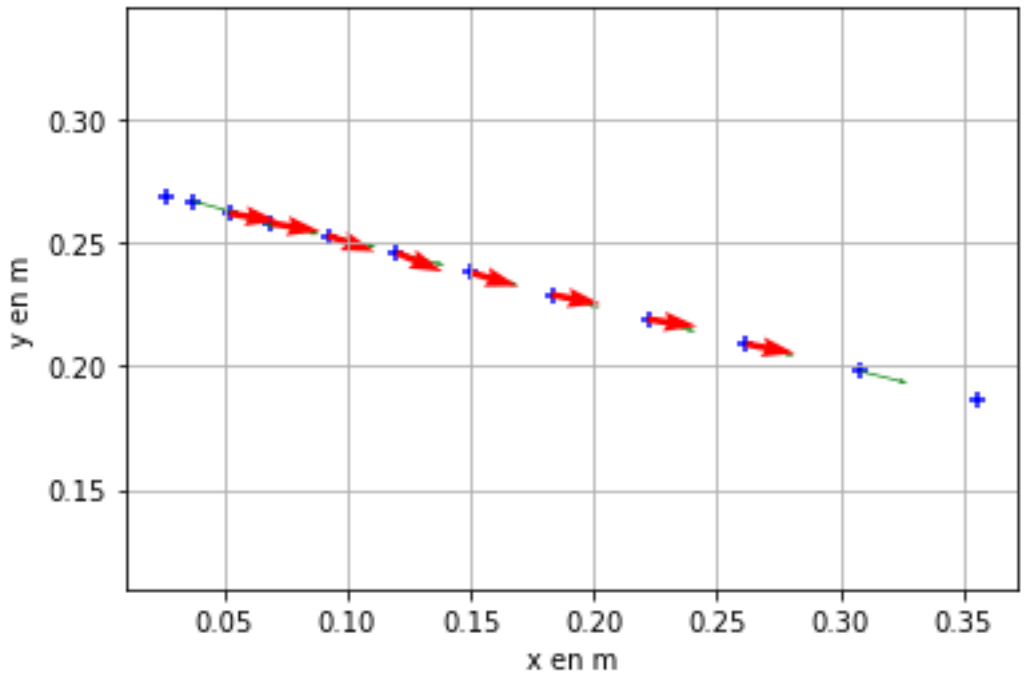
Vecteurs Vitesse :

```
import matplotlib.pyplot as plt
t=[0,0.049,0.1,0.145,0.191,0.238,0.284,0.33,0.376,0.422,0.469,0.517]
X=[0.026,0.037,0.052,0.069,0.092,0.12,0.15,0.183,0.222,0.262,0.308,0.356]
Y=[0.269,0.267,0.262,0.258,0.253,0.246,0.238,0.229,0.219,0.209,0.198,0.186] # compléter avec la liste Y des ordonnées y du point M

nombre_mesures = len(t) # nombre d'intervalles pour les calculs de vitesse

Vx=[]
Vy=[]

# Tracé de la trajectoire
plt.figure(1)  # Déclaration du graphique n°1, indispensable pour tracer plusieurs représentations graphiques
plt.scatter(X, Y, color = "blue", marker = '+')  # Tracé en nuage de points avec indications minimales
plt.xlabel("x en m")  # Étiquette de l'axe des abscisses
plt.ylabel("y en m")  # Étiquette de l'axe des ordonnées
plt.grid()  # Affichage de la grille
plt.axis("equal")  # Pour avoir un repère orthonormé

# Tracé des vecteurs vitesse
echelleV=1

for i in range(1, nombre_mesures-1) :
    delta_t = t[i+1]-t[i-1]  # formule pour déterminer l'intervalle de temps
    vx = (X[i+1]-X[i-1])/( delta_t)  # formule pour déterminer les coordonnées vx de la vitesse
    vy = (Y[i+1]-Y[i-1])/( delta_t) # formule pour déterminer les coordonnées vy de la vitesse
    Vx.append(vx),Vy.append(vy) # ajoute les coordonnées du vecteur vitesse à la liste pour le calcul de l'accélération
    plt.quiver(X[i], Y[i], vx*echelleV, vy*echelleV, color = "green",width=0.002)  # Représentation des vecteurs vitesse
    print("Au point",i,":(vx=",round(vx,3),"; vy=",round(vy,3),")") # affiche les coordonnées de la vitesse

plt.show()  # Affichage du graphique
```



Vecteurs accélération :

```
import matplotlib.pyplot as plt
t=[0,0.049,0.1,0.145,0.191,0.238,0.284,0.33,0.376,0.422,0.469,0.517]
X=[0.026,0.037,0.052,0.069,0.092,0.12,0.15,0.183,0.222,0.262,0.308,0.356]
Y=[0.269,0.267,0.262,0.258,0.253,0.246,0.238,0.229,0.219,0.209,0.198,0.186]
nombre_mesures = len(t) # nombre d'intervalles pour les calculs de vitesse
Vx,Vy=[],[]
# Tracé de la trajectoire
plt.figure(1)  # Déclaration du graphique n°1, indispensable pour tracer plusieurs représentations graphiques
plt.scatter(X, Y, color = "blue", marker = '+')  # Tracé en nuage de points avec indications minimales
plt.xlabel("x en m")  # Étiquette de l'axe des abscisses
plt.ylabel("y en m")  # Étiquette de l'axe des ordonnées
plt.grid()  # Affichage de la grille
plt.axis("equal")  # Pour avoir un repère orthonormé
echelleV=1
for i in range(1, nombre_mesures-1) :
    delta_t = t[i+1]-t[i-1]  # formule pour déterminer l'intervalle de temps
    vx = (X[i+1]-X[i-1])/( delta_t)  # formule pour déterminer les coordonnées vx de la vitesse
    vy = (Y[i+1]-Y[i-1])/( delta_t) # formule pour déterminer les coordonnées vy de la vitesse
    Vx.append(vx),Vy.append(vy)
    plt.quiver(X[i], Y[i], vx*echelleV, vy*echelleV, color = "green",width=0.002)
    print("Au point",i,":(vx=",round(vx,3),"; vy=",round(vy,3),")") # affiche les coordonnées de la vitesse

    # Tracé des vecteurs accélération
nombre_mesures = len(Vx) # nombre d'intervalles pour les calculs d'accélération
echellea=1
for i in range(1, nombre_mesures-1) :
    delta_t = t[i+2]-t[i]  # la vitesse n'est calculé qu'au point 1 pas au point 0
    ax = (Vx[i+1]-Vx[i-1])/delta_t  # formule pour déterminer les coordonnées ax de l'accélération
    ay = (Vy[i+1]-Vy[i-1])/delta_t # formule à compléter pour déterminer les coordonnées ay de l'accélération

    plt.quiver(X[i+1], Y[i+1], ax*echellea, ay*echellea, color = "red")  # Représentation des vecteurs vitesse
    print("Au point",i+1,":(ax=",round(ax,3),"; ay=",round(ay,3),")") # affiche les coordonnées de l'accélération

plt.show()  # Affichage du graphique
```



Au point 1 :(vx= 0.26 ; vy= -0.07)
Au point 2 :(vx= 0.333 ; vy= -0.094)
Au point 3 :(vx= 0.44 ; vy= -0.099)
Au point 4 :(vx= 0.548 ; vy= -0.129)
Au point 5 :(vx= 0.624 ; vy= -0.161)
Au point 6 :(vx= 0.685 ; vy= -0.185)
Au point 7 :(vx= 0.783 ; vy= -0.207)
Au point 8 :(vx= 0.859 ; vy= -0.217)
Au point 9 :(vx= 0.925 ; vy= -0.226)
Au point 10 :(vx= 0.989 ; vy= -0.242)

Au point 2 :(ax= 1.87 ; ay= -0.301)
Au point 3 :(ax= 2.363 ; ay= -0.388)
Au point 4 :(ax= 1.98 ; ay= -0.671)
Au point 5 :(ax= 1.467 ; ay= -0.599)
Au point 6 :(ax= 1.728 ; ay= -0.492)
Au point 7 :(ax= 1.89 ; ay= -0.354)
Au point 8 :(ax= 1.545 ; ay= -0.21)
Au point 9 :(ax= 1.406 ; ay= -0.266)