Chute d'une balle lachée depuis un vélo

Doc. 8,4

A Objectif

L'objectif de cette activité est l'étude expérimentale du mouvement d'une balle lâchée depuis un vélo dans différents référentiels : le référentiel terrestre et un référentiel lié au vélo.

B Partie expérimentale

B.1 Enregistrement des positions de la balle

- Charger la vidéo, nommée « `velo_et_balle.mp4` », située sur classroom, dans le logiciel mecachrono.
- Sélectionner « Nombre d'images par seconde de la vidéo : 25 » et « Nombre d'images entre deux échantillonnages : 1 ».
- Faire défiler la vidéo image par image, jusqu'à l'instant où la balle quitte la main du cycliste.
- Placer l'origine du repère sur la balle.
- Définir l'échelle en utilisant la règle à l'écran : *la distance qui sépare les centres des deux roues est égale à* $1{,}08\,\text{m}$.
- Cliquer sur les différentes positions de la balle, jusqu'à la date $0{,}64\,\text{s}$.
- Sélectionner l'onglet « Tableau de valeurs » et exporter les données sous une forme directement exploitable sous Python.
- Renommer `x_b` et `y_b` les variables dans le code.

B.2 Enregistrement de la position du centre de la roue du vélo

- Effacer toutes les positions de la balle et revenir à la première image de la section précédente.
- Cliquer sur les différentes positions du centre de la roue de devant, jusqu'à la date $0{,}64\,\text{s}$.
- Sélectionner l'onglet « Tableau de valeurs » et exporter les données sous une forme directement exploitable sous Python.

  Remarque. Seule l'abscisse x est intéressante.

- Renommer `x_v` la variable dans le code.

C Exploitation

C.1 Étude du mouvement de la balle par rapport au référentiel terrestre

Q 1. Insérer, dans le bloc `Q1`, les listes contenant les dates t et les valeurs des coordonnées x_b, y_b et x_v.

Solution :

```
t = [ 0, 0.04, 0.08, 0.12, 0.16, 0.2, 0.24, 0.28, 0.32, 0.36,
0.4, 0.44, 0.48, 0.52, 0.56]

x_b = [0, 0.197305, 0.389419, 0.591916, 0.789222, 0.991719,
1.18902, 1.37075, 1.56806, 1.76536, 1.95229, 2.14959, 2.33132,
2.51824, 2.70516]

y_b = [0.00259612, -0.0545186, -0.116826, -0.210286,
-0.314131, -0.42836, -0.573743, -0.713934, -0.89047,
-1.0722, -1.26431, -1.48239, -1.71085, -1.96007, -2.21449]

x_v = [0.612685, 0.820375, 1.02287, 1.22537, 1.43306, 1.63556,
1.83806, 2.04055, 2.24305, 2.43516, 2.64285, 2.84016, 3.03227,
3.23477]
```

Q 2. On cherche à tracer l'évolution des coordonnées x_b, y_b, de la position de la cours du temps t, par rapport au référentiel terrestre. Compléter le bloc Q2.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t, x_b, 'o', label="$x_b$ (m)")
plt.plot(t, y_b, 'o', label="$y_b$ (m)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

Q 3. Caractériser les mouvements de la balle selon les axes (Ox) et (Oy) par rapport au référentiel terrestre.

Solution : Le mouvement horizontal est uniforme, par rapport au référentiel terrestre, puisque la balle parcourt la même distance pendant des durées égales.

Le mouvement vertical est accéléré, par rapport au référentiel terrestre, puisque la balle parcourt des distances de plus en plus grandes pendant des durées égales.

Q 4. On cherche à modéliser, à partir des réponses à la question 3, l'évolution de l'abscisse x_b au cours du temps. Compléter le bloc de code Q3.
Les prédictions étaient-elles correctes ?

Solution :

```
# Fonction modèle
def modele(x, a, b):
    return a * x + b

# Détermination des paramètres optimaux
popt, pcov = curve_fit(modele, t, x_b)
a_mod = popt[0]
b_mod = popt[1]

# Résultat de la modélisation
t_mod = np.linspace(min(t), max(t), 100)
x_b_mod = modele(t_mod, a_mod, b_mod)

# Tracé
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t, x_b, 'o')
plt.plot(t_mod, x_b_mod)
plt.xlabel("t (s)")
plt.ylabel("$x_b$ (m)")
```

```
plt.show()

print("Paramètres de la modélisation : {}, {}".format(a_mod, b_mod))
```

Q 5. On cherche à modéliser, à partir des réponses à la question 3, l'évolution de l'ordonnée y_b au cours du temps. Compléter le bloc de code Q4.
Les prédictions étaient-elles correctes ?

Solution :

```
# Fonction modèle
def modele(x, a, b, c):
    return a * x**2 + b * x + c

# Détermination des paramètres optimaux
popt, pcov = curve_fit(modele, t, y_b)
a_mod = popt[0]
b_mod = popt[1]
c_mod = popt[2]

# Résultat de la modélisation
t_mod = np.linspace(min(t), max(t), 100)
y_b_mod = modele(t_mod, a_mod, b_mod, c_mod)

# Tracé
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t, y_b, 'o')
plt.plot(t_mod, y_b_mod)
plt.xlabel("t (s)")
plt.ylabel("$y_b$ (m)")
plt.show()

print("Paramètres de la modélisation : {}, {}, {}".format(a_mod, b_mod, c_mod))
```

Q 6. On cherche à afficher la trajectoire de la balle par rapport au référentiel terrestre en utilisant les coordonnées $x_{b,mod}$ et $y_{b,mod}$. Compléter le bloc de code Q5.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(x_b_mod, y_b_mod, '-', label="$y_b$ (m)")
plt.xlabel("$x_v$ (m)")
plt.legend()
plt.show()
```

Q 7. Nommer la trajectoire de la balle par rapport au référentiel terrestre.

Solution : La trajectoire est une parabole.

Q 8. On cherche à construire les listes des valeurs des composantes v_{xb} et v_{yb} de la vitesse $\vec{v}_b$ par rapport au référentiel terrestre. Compléter le bloc de code Q6.

Solution :

```
vx_b = [0] * len(x_b_mod)  # Que fait-on ?
vy_b = [0] * len(y_b_mod)  # Que fait-on ?

for i in range(1, len(t_mod) - 1):
    vx_b[i] = ( x_b_mod[i + 1] - x_b_mod[i - 1]) / ( t_mod[i + 1] - t_mod[i - 1])
    vy_b[i] = ( y_b_mod[i + 1] - y_b_mod[i - 1]) / ( t_mod[i + 1] - t_mod[i - 1])
```

Q 9. On cherche à afficher l'évolution des composantes v_{xb} et v_{yb} de la vitesse $\vec{v}_b$. Compléter le bloc de code Q7.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[1:-1], vx_b[1:-1], '-', label="$v_{xb}$ (m)")
plt.plot(t_mod[1:-1], vy_b[1:-1], '-', label="$v_{yb}$ (m)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

Q 10. On cherche à construire les listes des valeurs des composantes a_{xb} et a_{yb} de l'accélération $\vec{a}_b$ par rapport au référentiel terrestre. Compléter le bloc de code Q8.

Solution :

```
ax_b = [0] * len(t_mod)  # Que fait-on ?
ay_b = [0] * len(t_mod)  # Que fait-on ?

for i in range(2, len(t_mod) - 2):
    ax_b[i] = ( vx_b[i + 1] - vx_b[i - 1]) / ( t_mod[i + 1] - t_mod[i - 1])
    ay_b[i] = ( vy_b[i + 1] - vy_b[i - 1]) / ( t_mod[i + 1] - t_mod[i - 1])
```

Q 11. On cherche à afficher l'évolution des composantes a_{xb} et a_{yb} de l'accélération $\vec{a}_b$. Compléter le bloc de code Q9.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[2:-2], ax_b[2:-2], '-', label="$a_{xb}$ (m)")
plt.plot(t_mod[2:-2], ay_b[2:-2], '-', label="$a_{yb}$ (m)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

C.2 Étude du mouvement du centre de la roue par rapport au référentiel terrestre

Q 12. On cherche à caractériser l'évolution de x_v, position du centre de la roue, au cours du temps t, par rapport au référentiel terrestre. Compléter le bloc de code Q10.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t, x_v, 'o', label="$x_v$ (m)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

Q 13. Caractériser le mouvement du centre de la roue par rapport au référentiel terrestre.

Solution : Le centre de la roue parcourt des distances égales pendant des durées égales par rapport au référentiel terrestre ; son mouvement est rectiligne et uniforme.

Q 14. On cherche à modéliser, à partir de la réponse à la question 13, l'évolution de l'abscisse x_v au cours du temps. Compléter le bloc de code Q11.
La prédiction était-elle correcte ?

Solution :

```
# Fonction modèle
def modele(x, a, b):
    return a * x + b

# Détermination des paramètres optimaux
popt, pcov = curve_fit(modele, t, x_v)
a_mod = popt[0]
b_mod = popt[1]

# Résultat de la modélisation
t_mod = np.linspace(min(t), max(t), 100)
x_v_mod = modele(t_mod, a_mod, b_mod)

# Tracé
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t, x_v, 'o')
plt.plot(t_mod, x_v_mod)
plt.xlabel("t (s)")
plt.ylabel("$x_v$ (m)")
plt.show()

print("Paramètres de la modélisation : {}, {}".format(a_mod, b_mod))
```

C.3 Mouvement de la balle par rapport au vélo

Q 15. On souhaite déterminer la trajectoire de la balle dans un référentiel lié au vélo. Compléter le bloc de code Q12.

Solution :

```
x_B = [0] * len(x_b_mod)  # Que fait-on ?
for i in range(len(x_b_mod)):
    x_B[i] = x_b_mod[i] - x_v_mod[i]

y_B = y_b_mod

plt.figure(figsize=(8, 6), dpi=100)
plt.plot(x_B, y_B, '-')
plt.xlabel("$x_B$ (m)")
plt.ylabel("$y_B$ (m)")
plt.xlim(-3,0)
plt.show()
```

Q 16. Nommer la trajectoire de la balle par rapport au vélo.

Solution : Par rapport au vélo, la trajectoire est une droite verticale.

Q 17. On cherche à construire les listes des valeurs des composantes v_{xB} et v_{yB} de la vitesse $\vec{v}_B$ par rapport à un référentiel lié au vélo. Compléter le bloc de code Q13.

Solution :

```
vx_B = [0] * len(t_mod)   # Que fait-on ?
vy_B = [0] * len(t_mod)   # Que fait-on ?

for i in range(1, len(t_mod) - 1):
    vx_B[i] = (x_B[i + 1] - x_B[i - 1]) / (t_mod[i + 1] - t_mod[i - 1])
    vy_B[i] = (y_B[i + 1] - y_B[i - 1]) / (t_mod[i + 1] - t_mod[i - 1])
```

Q 18. On cherche à afficher l'évolution des composantes v_{xB} et v_{yB} de la vitesse $\vec{v}_B$. Compléter le bloc de code Q14.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[1:-1], vx_B[1:-1], '-', label="$v_{xB}\ (m/s)$")
plt.plot(t_mod[1:-1], vy_B[1:-1], '-', label="$v_{yB}\ (m/s)$")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

Q 19. On cherche à construire les listes des valeurs des composantes a_{xB} et a_{yB} de l'accélération $\vec{a}_B$ par rapport à un référentiel lié au vélo. Compléter le bloc de code Q15.

Solution :

```
ax_B = [0] * len(t_mod)   # Que fait-on ?
ay_B = [0] * len(t_mod)   # Que fait-on ?

for i in range(2, len(t_mod) - 2):
    ax_B[i] = (vx_B[i + 1] - vx_B[i - 1]) / (t_mod[i + 1] - t_mod[i - 1])
    ay_B[i] = (vy_B[i + 1] - vy_B[i - 1]) / (t_mod[i + 1] - t_mod[i - 1])
```

Q 20. On cherche à afficher l'évolution des composantes a_{xB} et a_{yB} de l'accélération $\vec{a}_B$. Compléter le bloc de code Q16.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[2:-2], ax_B[2:-2], '-', label="$a_{xB}\ (m/s^2)$")
plt.plot(t_mod[2:-2], ay_B[2:-2], '-', label="$a_{yB}\ (m/s^2)$")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

C.4 Étude de l'évolution des différentes formes d'énergie

Remarque. Dans la suite de ce document, on considère des énergies massiques.

C.4.1 Par rapport au référentiel terrestre

Q 21. On cherche à construire la liste des valeurs de l'énergie cinétique, par rapport au référentiel terrestre. Compléter le bloc de code Q17.

Solution :

```
Ec = [0] * len(t_mod)  # Que fait-on ?

for i in range(1, len(t_mod) - 1):
    Ec[i] = 0.5 * (vx_b[i] ** 2 + vy_b[i] ** 2)
```

Q 22. On cherche à construire la liste des valeurs de l'énergie potentielle de pesanteur, par rapport au référentiel terrestre. Compléter le bloc de code Q18.

Remarque. Faire en sorte que l'énergie potentielle de pesanteur soit nulle pour la position finale de la balle.

Solution :

```
Epp = [0] * len(t_mod)
g = 9.81

for i in range(len(t_mod)):
    Epp[i] = g * (y_b_mod[i] - y_b[-1])
```

Q 23. On cherche à construire la liste des valeurs de l'énergie mécanique, par rapport au référentiel terrestre. Compléter le bloc de code Q19.

Solution :

```
Em = [0] * len(t_mod)

for i in range(len(t_mod)):
    Em[i] = Ec[i] + Epp[i]
```

Q 24. On souhaite afficher l'évolution des différentes formes d'énergie par rapport au référentiel terrestre. Compléter le bloc de code Q20.

Solution :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[1:-1], Ec[1:-1], '-', label="$E_c$ (J)")
plt.plot(t_mod[1:-1], Epp[1:-1], '-', label="$E_{pp}$ (J)")
plt.plot(t_mod[1:-1], Em[1:-1], '-', label="$E_{m}$ (J)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

C.4.2 Par rapport à un référentiel lié au vélo

Q 25. Reprendre les questions précédentes dans un référentiel lié au vélo.

Solution :

```
# Énergie cinétique
Ec_B = [0] * len(t_mod)  # Que fait-on ?

for i in range(1, len(t_mod) - 1):
    Ec_B[i] = 0.5 * (vx_B[i] ** 2 + vy_B[i] ** 2)

# Énergie potentielle de pesanteur
Epp_B = [0] * len(t_mod)
g = 9.81

for i in range(len(t_mod)):
    Epp_B[i] = g * (y_B[i] - y_B[-1])

# Énergie mécanique
Em_B = [0] * len(t_mod)

for i in range(len(t_mod)):
    Em_B[i] = Ec_B[i] + Epp_B[i]

plt.figure(figsize=(8, 6), dpi=100)
plt.plot(t_mod[1:-1], Ec_B[1:-1], '-', label="$E_c$ (J)")
```

```
plt.plot(t_mod[1:-1], Epp_B[1:-1], '-', label="$E_{pp}$ (J)")
plt.plot(t_mod[1:-1], Em_B[1:-1], '-', label="$E_{m}$ (J)")
plt.xlabel("t (s)")
plt.legend()
plt.show()
```

C.4.3 Synthèse

Q 26. Quelles similarités et différences présentent les courbes des différentes formes d'énergie obtenues dans les sections C.4.1 et C.4.2 ?

Solution : Dans les deux situations, on observe que l'énergie mécanique reste constante : les frottements sont négligeables. Cependant, les valeurs des énergies diffèrent. Il est particulièrement intéressant d'analyser ces valeurs à l'instant $t = 0$: dans le référentiel du vélo, toute l'énergie est sous forme d'énergie potentielle de pesanteur (étant donné que la bille est relâchée sans vitesse initiale), tandis que dans le référentiel terrestre, elle se divise entre énergie potentielle d'interaction et énergie cinétique.

Q 27. Quelle conclusion sur l'énergie peut-on déduire de l'étude réalisée dans cette section ?

Solution : L'étude illustre le fait que certaines formes d'énergie (cinétique, potentielle d'interaction, etc.) ne sont pas absolues mais dépendent du référentiel d'étude. *Dans tous les cas, les variations d'énergie sont indépendantes de ce référentiel.*