Activité numérique

python

3 Simulation d'un lancer de poids

Script à compléter

Fichiers Python
Scripts à compléter
Fiches d'accompagnement
hatier-clic.fr/pc1244

L'objectif de cette activité est de simuler numériquement la trajectoire d'un corps soumis à son poids mais aussi aux actions de l'air (frottements et vent). Dans ce cas, il nous est impossible d'obtenir l'expression mathématique de la trajectoire de l'objet. L'utilisation de la simulation numérique est donc justifiée.

Prérequis théoriques

Une égalité vectorielle implique les égalités des projections vectorielles des vecteurs :

si les vecteurs $\vec{a}$, $\vec{b}$, $\vec{c}$ vérifient $\vec{a} = \vec{b} + \vec{c}$, alors $\begin{cases} a_x = b_x + c_x \\ a_y = b_y + c_y \end{cases}$.

La question 1a nécessite de compléter le programme « 11_mouvement.py » :
- en complétant les coordonnées du vecteur somme des forces (lignes 49 et 50) ;
- en complétant les coordonnées de la variation du vecteur vitesse (lignes 52 et 53) ;
- en complétant les coordonnées du vecteur vitesse à l'instant $t + \Delta t$ (lignes 55 et 56).

Ces compléments sont à déterminer en utilisant le doc. 2 (pour la prise en compte de l'ensemble des forces agissant sur le système) et le doc. 3 (pour l'utilisation de la deuxième loi de Newton pour déterminer le mouvement à l'instant $t + \Delta t$ à partir de l'instant t).

Ensuite, il reste à exécuter le programme pour obtenir la trajectoire de la boule. On utilise le graphique pour déterminer la portée de la boule (les coordonnées du curseur apparaissent dans le coin inférieur droit).

La question 1b nécessite de modifier le programme « 11_mouvement.py » pour que celui-ci calcule directement la portée du lancer en utilisant la fonction Python `max` (ligne 69).

Ensuite, il reste à exécuter le programme pour obtenir la trajectoire de la boule et, dans la fenêtre d'exécution du programme Python, les indications des paramètres de lancer de la boule, ainsi que sa portée.

La question 2a nécessite de modifier le programme « 11_mouvement.py » pour prendre en compte une force de frottement de l'air et une force due au vent (lignes 22 et 23), puis d'exécuter le programme pour déterminer la nouvelle portée du lancer et la comparer avec celle obtenue sans action de l'air.

La question 2b nécessite de modifier le programme « 11_mouvement.py » pour changer la valeur de l'angle de lancer α (ligne 21) et d'observer l'influence de ce paramètre sur la portée de tir. Plus précisément, il est demandé de chercher l'angle (au degré près) permettant d'obtenir la portée maximale de tir sans modifier les autres paramètres de lancer.

La question 3a nécessite de modifier le programme « 11_portee.py » pour introduire une boucle `for` dans le programme (ligne 34) pour calculer, pour chaque angle allant de 0° à 90°, la portée du tir. Il est à noter que, comme dans toute boucle ou autre instruction de test, le bloc des instructions de la boucle doit être décalé (en utilisant la touche tabulation), on parle aussi d'indentation (lignes 36 et 40).

Sans cette manipulation les instructions qui se trouvent sous la ligne 34 (les lignes 36 et 40) ne seront pas reconnues comme des instructions faisant parties de la boucle `for`.

Dans la question 3b, on exécute le programme « 11_portee.py » qui, en plus d'afficher un diagramme représentant la portée en fonction de l'angle de lancer, affiche, dans la fenêtre d'exécution du programme Python, les indications des paramètres de lancer de la boule, ainsi que l'angle optimal et la portée correspondante. On peut ainsi comparer l'angle optimal obtenu avec l'angle obtenu à la question 2b.

La question 4a nécessite de modifier le programme « 11_portee.py » pour changer la valeur de la vitesse de lancer (ligne 22) et de déterminer la valeur de la vitesse de lancer permettant d'obtenir la portée maximale attendue (22,50 m).

La question 4b nécessite de modifier le programme « 11_portee.py » pour changer la valeur de la force du vent (ligne 24) sans modifier les autres paramètres et ainsi, déterminer la valeur de la portée maximale dans ces nouvelles conditions.

Script à compléter (programme « 11_mouvement.py »)

```
1  import numpy as np
2  import matplotlib.pylab as plt
3
4  print("")
5  print("*****************************")
6  print("* Mouvement d'un projectile *")
7  print("*****************************")
8  print("")
9
10 ### Données
11 h=2.2        # Altitude initiale de la boule (m)
12 v0=10        # Norme de la vitesse initiale (m/s)
13 m=4.00       # Masse du système (kg)
14 g=9.81       # Norme du champ de pesanteur (N/kg)
15 Np=100000    # Nombre de pas maximal de calcul
16 Dt=1E-2      # Pas de temps (s)
17
18 ########################################
19 ### Données à modifier (question 2) ###
20 ########################################
21 angle=...    # Angle de lancer au-dessus de l'horizontale (°)
22 k=...        # Coefficient de frottement k (N.s/m)
23 Tx=...       # Force du vent Tx (N)
24
25 ### Initialisation
26 ### Création de tableaux remplis de zéros
27 t=np.zeros(Np+1)
28 x=np.zeros(Np+1)
29 y=np.zeros(Np+1)
30 vx=np.zeros(Np+1)
31 vy=np.zeros(Np+1)
32 ### Position initiale de la boule
33 x[0]=0
34 y[0]=h
```

Modules

Le module `numpy` est nécessaire la création de toutes la matrices du programme et l'utilisation des fonctions mathématiques trigonométriques.

Le module `matplotlib` permet de tracer les graphiques.

Données du problème

On définit les constantes propres à un lancer, ainsi que des constantes propres à la simulation (pas de temps et nombre de points maximum). Certaines de ces données seront modifiées au cours de cette activité.

Initialisation des matrices

Les matrices sont initialement remplies avec des zéros.
Dans Python, il aurait été possible d'étendre au fur et à mesure les matrices pour ne pas les prédéfinir ainsi. Reste que les instructions de la boucle ne se seraient plus écrit sous la forme « A[p+1] = A[p] + B » mais sous la forme « A.append(A[p] + B) ».

Activité numérique

Chapitre 11, p. 244

```
### Coordonnées du vecteur vitesse initiale
vx[0]=v0*np.cos(angle/180*np.pi)
vy[0]=v0*np.sin(angle/180*np.pi)
### Pas de calcul
p=0

### Boucle de calcul du mouvement
while y[p]>=0:
       # Calcul de la date
       t[p+1]=t[p]+Dt
       ########################################
       ### À compléter : question 1a puis 2 ###
       ########################################
       ### Coordonnées de la somme des forces
       Fx=...
       Fy=...
       ### Coordonnées de la variation du vecteur vitesse
       Dvx=...
       Dvy=...
       ### Coordonnées du vecteur vitesse à l'instant t+Dt
       vx[p+1]= ...
       vy[p+1]= ...

       ### Ne pas modifier :
       ### Coordonnées de la position à l'instant t+Dt
       x[p+1]=x[p]+vx[p]*Dt
       y[p+1]=y[p]+vy[p]*Dt
       ### Avancement du pas de calcul
       p=p+1

##################################
#### À modifier : question 1b ####
### Détermination de la portée ###
##################################
portee=...        # Expression à modifier

### Affichage des données et de la portée
print("Données :")
print("- masse du système m =",m,"kg")
print("- hauteur de lancer h =",h,"m")
print("- norme de la vitesse initiale v0 =",v0,"m/s")
print("- angle de tir",angle,"°")
print("- coefficient de frottement k =",k,"N.s/m")
print("- force due au vent horizontal Tx =",Tx," N")
print("Portée du tir :",round(portee,2),"m")
print("")

### Tracé de la trajectoire du système
plt.plot(x[0:p],y[0:p],'-',lw=2)
plt.xlabel("x (en m)")
plt.ylabel("y (en m)")
plt.title("Trajectoire du système")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Coordonnées du vecteur $\vec{v}_0$

On définit les coordonnées du vecteur vitesse à l'instant initial.

La boucle

On utilise ici une boucle `while` qui permet d'arrêter les calculs après que la boule a touché le sol. (L'instant d'après, son ordonnée deviendra négative, puisque rien n'a été programmé pour simuler le sol et l'influence de celui-ci sur la boule.)
Dans cette boucle, on calcule tout d'abord les coordonnées du vecteur force totale $\vec{F}_{\text{tot}}$ puis les coordonnées de la variation du vecteur vitesse $\Delta\vec{v}$, pour en déduire les coordonnées du vecteur vitesse $\vec{v}$ et pour finir par les coordonnées de la position de boule $\overrightarrow{OM}$.

Incrémentation

On incrémente le pas d'itération `p`.

Détermination de la portée du tir

On utilise la fonction Python `max` pour déterminer la portée du lancer.

Communication des données et résultats

On utilise la fonction Python `print` pour qu'apparaissent, dans la fenêtre d'exécution du programme Python, les données du lancer, ainsi que la portée obtenue.

Tracé de la trajectoire

On trace la trajectoire obtenue.

Activité numérique

Chapitre 11, p. 244

Script à compléter (programme « 11_portee.py »)

```
1  import numpy as np
2  import matplotlib.pylab as plt
3
4  print("")
5  print("**************************")
6  print("* Portée d'un projectile *")
7  print("**************************")
8  print("")
9
10
11 ########################################################
12 ###  Création de la fonction "trajectoire_portee"  ###
13 ########################################################
14
15 def trajectoire_portee(m,g,k,h,Tx,v,angle):
16     #Nombre de pas maximum
17     Np=100000
18
19     #Pas de temps en seconde
20     Dt=1E-2
21
22     #Initialisation des coordonnées du système
23     t=np.zeros(Np+1)
24     x=np.zeros(Np+1)
25     y=np.zeros(Np+1)
26     vx=np.zeros(Np+1)
27     vy=np.zeros(Np+1)
28
29     #Position initiale de la particule
30     x[0]=0
31     y[0]=h
32
33     #Coordonnées du vecteur vitesse initiale au moment du lancer
34     vx[0]=v*np.cos(angle/180*np.pi)
35     vy[0]=v*np.sin(angle/180*np.pi)
36
37
38     ########################################################
39     #BOUCLE de mouvement de la particule
40     ########################################################
41
42     #La boucle
43     p=0
44     while y[p]>=0:
45
46         #La valeur de l'instant en cours
47         t[p+1]=t[p]+Dt
48
49         #Les coordonnées de la somme des forces appliquées au système :
50         Fx=0-k*vx[p]+Tx
51         Fy=-m*g-k*vy[p]
52
53         #Les coordonnées de la variation du vecteur vitesse :
54         Dvx=Fx/m*Dt
55         Dvy=Fy/m*Dt
56
57         #Les coordonnées du vecteur vitesse à l'instant t+Dt :
58         vx[p+1]=vx[p]+Dvx
59         vy[p+1]=vy[p]+Dvy
60
61         #Les coordonnées du système à l'instant t+Dt :
62         x[p+1]=x[p]+vx[p]*Dt
63         y[p+1]=y[p]+vy[p]*Dt
64
65         p=p+1
66
67     return max(x)
68
```

Modules

Le module `numpy` est nécessaire la création de toutes la matrices du programme et l'utilisation des fonctions mathématiques trigonométriques. Le module `matplotlib` permet de tracer les graphiques.

Fonction

Fonction `trajectoire_portee` qui contient les instructions du programme « 11_mouvement.py ».

Activité numérique

Chapitre 11, p. 244

```
### Données
h=2.2          # Altitude initiale de la boule (m)
m=4.00           # Masse du système (kg)
g=9.81           # Norme du champ de pesanteur (N/kg)
Np=100000      # Nombre de pas maximal de calcul
Dt=1E-2                # Pas de temps (s)

########################################
### Données à modifier (question 4) ###
########################################
v0=10          # Norme de la vitesse initiale (m/s)
k=0.1          # Coefficient de frottement k (N.s/m)
Tx=-10         # Force du vent Tx (N)

### Initialisation des tableaux
portee=np.zeros(91)
angle=np.zeros(91)

############################################
### À compléter et modifier : question 3a ###
############################################
### Indice qui sera modifié par la boucle for à introduire
n=0
### Angle de tir, à faire varier entre 0 et 90°
angle[n]=n
### Utilisation de la fonction "trajectoire_portee"
### figurant en début de programme
### pour déterminer les variations de x
portee[n]=trajectoire_portee(m,g,k,h,Tx,v0,angle[n])

### Identification de la portee maximale (ne pas modifier)
porteemax=int(100*portee[np.where(portee==np.max(portee))])/100
### Identification de l'angle optimal de lancer
angleopt=int(angle[np.where(portee==np.max(portee))])

### Affichage des données et de la portée
print("Données :")
print("- masse du système m =",m,"kg")
print("- hauteur de lancer h =",h,"m")
print("- norme de la vitesse initiale v0 =",v0,"m/s")
print("- coefficient de frottement k =",k,"N.s/m")
print("- force due au vent horizontal Tx =",Tx," N")
print("Portée du tir maximale :",porteemax,"m")
print("pour un angle de tir",angleopt,"°")
print("")

### Tracé de la portée en fonction de l'angle de tir
fig=plt.figure()
ax=fig.add_subplot(111,polar=True)
c=ax.plot(angle/180*np.pi,portee)
ax.set_thetamin(0)
ax.set_thetamax(90)
plt.title("Portée en fonction de l'angle de tir")
plt.show()
```

Données du problème

On définit les constantes propres à un lancer, ainsi que des constantes propres à la simulation (pas de temps et nombre de points maximum). Certaines de ces données seront modifiées au cours de cette activité.

Initialisation des matrices

Les matrices sont initialement remplies avec des zéros. Dans Python, il aurait été possible d'étendre au fur et à mesure les matrices pour ne pas les prédéfinir ainsi. On aurait alors remplacé les instructions `angle[n]=…` et `portee[n]=…` par `angle.append(…)` et `portee.append(…)`.

La boucle

On utilise ici une boucle `for` qui permet de remplir les matrices des angles et portées pour un indice *n* prenant les valeurs entières de 0 à 90.
La détermination de la portée se réalise en appelant la fonction `portee` qui se trouve dans le fichier « trajectoire.py » importé au début du programme. Cette fonction reprend l'ensemble du programme vu dans le programme « 11_mouvement.py ».

Communication des données et résultats

On utilise la fonction Python `print` pour qu'apparaissent, dans la fenêtre d'exécution du programme Python, les données du lancer, ainsi que la portée obtenue pour l'angle optimal.

Tracé de la portée

On trace un diagramme représentant la portée en fonction de l'angle de lancer.