2 Deuxième loi de Kepler

Programme à compléter

L'objectif de cette activité est d'exploiter des données satellitaires pour tester la deuxième loi de Kepler.

Fichiers Python
Programme à compléter
Fiche d'accompagnement
hatier-clic.fr/pct377

Prérequis théoriques

- Latitude et longitude

Apports théoriques supplémentaires de l'activité

- Formule de Héron pour le calcul de l'aire d'un triangle

La question **1** consiste à collecter, sur le site www.n2yo.com, les données concernant le satellite choisi sur le site de suivi : dates d'observation, altitudes au-dessus du sol, latitude, longitude.
Il faut avoir en tête que les positions du satellite sont des données *calculées* par le site, et non observées en temps réel.
Il est en effet très rare de trouver des données satellitaires ou astronomiques où l'on ait les positions d'un satellite ou d'un objet céleste en fonction du temps.
Quand on en trouve, il s'agit en général de données peu exploitables à moins de solides connaissances en matière de systèmes de coordonnées astronomiques.
Le choix a ainsi été fait, pour traiter le programme, de réaliser une expérience numérique factice puisqu'elle s'appuie sur des données calculées.
Néanmoins, les positions réelles des satellites sont régulièrement observées pour remettre à jour le jeu de données.

La question **2** implique des modifications du programme pour calculer les grandeurs demandées. Elle peut se faire pendant la collecte de données (mais il faut veiller au temps qui s'écoule pour, si possible, ne pas rater de dates).

Le logiciel calcule les coordonnées cartésiennes du satellite en fonction du temps dans le référentiel géocentrique.

Il faut donc, pour calculer la distance entre une position et la suivante, utiliser le théorème de Pythagore à trois dimensions.

Si M$(x,\ y,\ z)$ est la position à la date t et M'$(x'\ ,y'\ ,z')$ la position à la date t', la distance entre ces deux points est :

$$\mathrm{MM'} = \sqrt{(x'-x)^2 + (y'-y)^2 + (z'-z)^2}$$

La vitesse v se calcule simplement comme :

$$v(t) = \frac{\mathrm{MM'}}{t'-t}$$

La formule de Héron permet de calculer l'aire $S(t)$ du triangle balayé par le satellite entre la date t, où il est à la distance r du centre de la Terre, et la date t', où il est à la distance r' : il s'agit de l'aire d'un triangle dont les côtés ont pour longueurs r, r' et MM'.

Enfin, la vitesse aréolaire s se calcule à l'aide de l'expression fournie.
Le programme donne les courbes de r et de v en fonction du temps : on voit que ces grandeurs sont variables. Il trace aussi s en fonction du temps, qui est constante, ce qui valide la deuxième loi de Kepler.

Programme à compléter

```
from math import *
import numpy as np
import matplotlib.pylab as plt

print
print("**************************")
print("* Deuxieme loi de Kepler *")
print("**************Hatier 2020*")
print

#####################################################
###                Donnees a modifier             ###
### Attention, le separateur decimal est le point ###
#####################################################
### Ici : satellite DSX
## t en min
t=[...]
## latitude en degres
lat=[...]
## longitude en degres
lon=[...]
## altitude en km
h=[...]

### Ne pas modifier
## Rayon terrestre en km
Rt=6378.0
## Jour sideral en min
Js=1436.066
## Vitesse angulaire en degres/min
w=360/Js
## nombre de positions
n=len(t)

## initialisations
r=np.zeros(n)
loncr=np.zeros(n)
latr=np.zeros(n)
x=np.zeros(n)
y=np.zeros(n)
z=np.zeros(n)
d=np.zeros(n-1)
v=np.zeros(n-1)
S=np.zeros(n-1)
s=np.zeros(n-1)
tt=np.zeros(n-1)

## positions dans referentiel geocentrique en km
for i in range(n):
## rayon du satellite
        r[i]=h[i]+Rt
## longitude corrigee de la rotation propre, en rad
        loncr[i]=(lon[i]+w*t[i])/180*pi
## latitude en rad
        latr[i]=lat[i]/180*pi
## coordonnees en km
        x[i]=r[i]*cos(latr[i])*cos(loncr[i])
        y[i]=r[i]*cos(latr[i])*sin(loncr[i])
        z[i]=r[i]*sin(latr[i])

## mouvement
for i in range(n-1):
## Temps sans derniere mesure
        tt[i]=t[i]

```

Modules

Le module `math` est nécessaire pour les fonctions trigonométriques et la valeur de π.
Le module `numpy` permet des opérations sur les listes.
Le module `matplotlib` fournit les fonctionnalités graphiques.

Données à modifier

Question **1** : les différentes dates t, et les latitudes, longitudes et altitudes des satellites étudiés à ces dates.

Données à ne pas modifier

- Le rayon terrestre, qui sera utilisé dans le calcul de la distance entre le satellite et le centre de la Terre.
- Le jour sidéral et la vitesse angulaire sont, quant à eux, utilisés dans la transformation des coordonnées pour passer en référentiel géocentrique.

Attention Le séparateur décimal est le point, pas la virgule.

Initialisations (ne pas modifier)

Les différentes listes utilisées sont créées, avec les tailles adaptées, ne contenant pour l'instant que des zéros.

Calcul des coordonnées dans le référentiel géocentrique (ne pas modifier)

La liste `r` contient les distances entre le centre de la Terre et le satellite.
La liste `latr` contient les latitudes, transformées de degrés en radians pour pouvoir être utilisées ultérieurement sans se poser de questions.
La liste `loncr` contient les longitudes corrigées de la rotation terrestre pendant l'étude. (En effet, les coordonnées fournies par le site sont relatives au référentiel terrestre.)

```
### Fonction de calcul de l'aire
### avec formule de Heron
def aire(a,b,c):
        p=(a+b+c)/2
        S=(p*(p-a) *(p-b)*(p-c))**0.5
        return S

####################
#### A COMPLETER ###
####################
for i in range(n-1):
## Distance parcourue entre i et i+1, en km
        d[i]=...
## vitesse en km/min
        v[i]=...
## aire balayee en km2
        S[i]=...
## vitesse areolaire en km2/min
        s[i]=...

### Ne pas modifier

tmax=max(tt)+2
rmax=max(r)*1.1
vmax=max(v)*1.1
Smax=max(S)*1.1
smax=max(s)*1.1

fig=plt.figure(1,figsize=(4,9))
fig.subplots_adjust(bottom=0.085, left=0.225, top = 0.975, right=0.975)
fig.add_subplot(3,1,1)
plt.plot(t,r,"+")
##plt.title("Rayon en fonction du temps")
plt.xlabel("temps en min")
plt.ylabel("rayon en km")
plt.xlim(0,tmax)
plt.ylim(0,rmax)
plt.grid(True)
fig.add_subplot(3,1,2)
plt.plot(tt,v,"+")
##plt.title("Vitesse en fonction du temps")
plt.xlabel("temps en min")
plt.ylabel("vitesse en km/min")
plt.xlim(0,tmax)
plt.ylim(0,vmax)
plt.grid(True)
fig.add_subplot(3,1,3)
plt.plot(tt,s,"+")
##plt.title("Vitesse areolaire en fonction du temps")
plt.xlabel("temps en min")
plt.ylabel("vitesse areolaire en km^2/min")
plt.xlim(0,tmax)
plt.ylim(0,smax)
plt.grid(True)

plt.show()
```

Formule de Héron (ne pas modifier)

Une fonction prenant les trois côtés du triangle et calculant l'aire à l'aide de la formule de Héron est créée. L'élève n'est pas obligé de l'utiliser et peut faire le calcul directement.

Calculs demandés (question 2) (à modifier)

La distance entre deux positions successives est calculée à l'aide d'un théorème de Pythagore à trois coordonnées.
La vitesse en un point donné est calculée comme la vitesse moyenne entre ce point et le suivant.
L'aire balayée entre un point et le suivant est calculée à l'aide de la formule de Héron. Les côtés du triangle sont donnés sur le doc. 4.
La vitesse aréolaire (qu'il ne serait pas indispensable de calculer si l'on pouvait garantir que les mesures sont prises à intervalles de temps réguliers) s'obtient à l'aide de l'aire balayée.

Tracés (à ne pas modifier)

Le programme réalise une figure avec trois graphiques, représentant le rayon de l'orbite, la vitesse linéaire et la vitesse aréolaire en fonction du temps.