3eme-loi-de-kepler-2-prof

June 15, 2023

La 3ème Loi de Képler : 2 - Caractéristiques des orbites

On ne s'intéresse plus qu'aux mouvements dans le référentiel h'liocentrique (les trajectoires des astres qui système solaire sont toutes de ellipses. Le but de cette activité est de déterminer la valeur du demi-grand axe et de la période de révolution de l'orbite d'une planète.

Les données utilisées sont issues du portail de l'Observatoire de Paris. Les éphémérides des planètes telluriques ont été regroupées dans des fichiers de la forme 'Planete_H.csv' ou 'Planete_G.csv' (sans accent) selon le référentiel héliocentrique ou géocentrique. La fonction ***positions*** *permet de retranscrire ces données dans un tableau exploitable par le langage PYTHON. En indice* ***[0]*** *on aura les dates, en indice* ***[1]*** *les abscisses, en indice* ***[2]*** *les ordonnées et en indice* ***[3]*** *la distance au repère (Coordonnées cartésiennes dans le plan de l'écliptique du référentiel).*

[1]:
```
# -*- coding: utf-8-*-

# Importation des bibliothèques
import numpy as np
import matplotlib.pyplot as plt
import csv
import statistics as stat
```

La fonction positions établit un tableau de valeurs indiquant chaque jour l'abscisse, l'ordonnée, la distance au référentiel pour un astre.

[2]:
```
def positions(fichier):
    valeurs = []
    Mesures = open(fichier,encoding="utf8", errors='ignore')
    csv_astre = csv.reader(Mesures,delimiter=";")      # On ouvre le fichier csv
    for row in csv_astre:     # On remplit le tableau à partir du fichier csv
        valeurs.append(row)
    valeurs = [list(map(float, x)) for x in valeurs]    # On transforme les␣
 ↪valeurs en flottant (en réels)
    return valeurs
```

II. Détermination de la valeur du demi-grand axe et de la période de l'orbite des planètes.

La fonction caracteristiques prend en argument un fichier de type 'Planete_H.csv' et retourne

la valeur du demi_grand axe a et de la période de révolution T. On a ainsi a = caracteristiques(fichier)[0] et aussi T = caracteristiques(fichier)[1]

[3]:
```
def caracteristiques(fichier):
    coordonnee = np.transpose(positions(fichier))
    aphelie, perihelie = max(coordonnee[3]), min(coordonnee[3]) # coordonnee[3]␣
 ↪est la liste des distances
    a = (aphelie + perihelie) / 2
    date_min = list(np.where(coordonnee[3] == perihelie))[0][0]
    date_max = list(np.where(coordonnee[3] == aphelie))[0][0]
    T = np.abs(float(date_max-date_min))*2
    return a,T
```

Question 6 : Que sont l'aphélie et la périhélie? Appliquer cette fonction en prenant la Terre comme argument (fichier Terre_H.csv). En quelles unités sont exprimées les résultats?

[4]:
```
caracteristiques('Terre_H.csv')
```

[4]:
```
(0.9999986323245, 368.0)
```

L'aphélie constitue le point où une planète est à la plus grande distance du Soleil.

Le périhélie constitue le point où une planète est à la distance minimale du Soleil.

a = 1.0 (a est exprimée en unités astronomiques)

T = 368 (T est exprimé en jour)

Question 7 : Recopier la fonction caractéristiques,quelles modifications apporter à la fonction **caracteristiques** afin que les valeurs calculées soient exprimées dans les unités du système international?

Il faut multiplier a par 149597870700 (valeur en mètre de l'UA) et T par 86400 (nombre de secondes dans 1 jour).

[5]:
```
def caracteristiques(fichier):
    coordonnee = np.transpose(positions(fichier))
    aphelie, perihelie = max(coordonnee[3]), min(coordonnee[3]) # coordonnee[3]␣
 ↪est la liste des distances
    a = (aphelie + perihelie) / 2 * 149597870700
    date_min = list(np.where(coordonnee[3] == perihelie))[0][0]
    date_max = list(np.where(coordonnee[3] == aphelie))[0][0]
    T = np.abs(float(date_max-date_min))*2 * positions(fichier)[0][0]* 86400
    return a,T
```

La fonction caracteriques_moyennes détermine pour une planète le couple (a,T) sur 10 intervalles de temps (étalés sur plusieurs siècles afin d'avoir des valeurs moyennes) en prenant comme argument les fichiers de la forme 'Planete_H.csv'

On a ainsi DemiGrand_axe = caracteristiques_moyennes(fichier)[0] qui correspond à une liste de 10 valeurs de a et aussi Periode = caracteristiques_moyennes(fichier)[1] qui correspond à une liste

de 10 valeurs de T

[6]:
```
def caracteristiques_moyennes(fichier):
    Planete = []
    DemiGrand_axe = []
    Periode = []
    ligne = positions(fichier)
    tranche = int(len(ligne)/10)
    for i in range(10):
        Planete.append([])
        for j in range(i*tranche,(i+1)*tranche):
            Planete[i].append(ligne[j])
        aphelie, perihelie = max(np.transpose(Planete[i])[3]), min(np.
 ↪transpose(Planete[i])[3]) # coordonnee[3] est la liste des distances
        a = (aphelie + perihelie) / 2 * 149597870700
        date_min = list(np.where(np.transpose(Planete[i])[3] ==␣
 ↪perihelie))[0][0]
        date_max = list(np.where(np.transpose(Planete[i])[3] == aphelie))[0][0]
        T = np.abs(float(date_max-date_min))*2 * positions(fichier)[0][0]* 86400
        DemiGrand_axe.append(a)
        Periode.append(T)
    return DemiGrand_axe, Periode
```

Question 8 : Comment à partir de l'aphélie et de la périhélie peut-on estimer la circularité d'une trajectoire elliptique?

Pour une trajectoire parfaitement circulaire aphélie et périhélie sont à la même distance du Soleil.

Question 9 : Déterminer la liste des valeurs de a et de T de l'orbite de la Terre sur les dix intervalles de temps à l'aide du fichier 'Terre_H.csv'.

[7]:
```
caracteristiques_moyennes('Terre_H.csv')[0]
```

[7]:
```
[149596937997.89203, 149601848384.34512, 149593830599.2868, 149596172976.7985,
149600618682.16617, 149595752306.3818, 149601962222.15384, 149597011248.18274,
149593012700.7764, 149601585950.82938]
```

[8]:
```
caracteristiques_moyennes('Terre_H.csv')[1]
```

[8]:
```
[31104000.0, 31795200.0, 31795200.0, 31104000.0, 31968000.0, 31622400.0,
31276800.0, 31795200.0, 31276800.0, 31622400.0]
```

Question 10 : Effectuer les moyennes et présentez le résultat de la mesure sous la forme X = moyenne(X) +/- incertitude-type en écriture scientifique.

T = (3.15 +/- 0.01)E+07 s

a = (1.50E+11 +/- 0.00001)E+11 m

La fonction estimation prend une liste en argument et retourne la moyenne des valeurs des cette liste en indice [0] ainsi que l'incertitude en indice [1].

[9]:
```
def estimation(liste):
    moyenne = stat.mean(liste)
    incertitude = stat.stdev(liste)/np.sqrt(len(liste))
    print('X = ',format(moyenne,'.2E'),' +/- ',format(incertitude,'.0E'))
    return moyenne,incertitude
```

Question 11 : Utiliser la fonction estimation avec en argument caracteristiques_moyennes(fichier)[0] ou [1] pour retrouver les résultats de la Question 9

[10]:
```
estimation(caracteristiques_moyennes('Terre_H.csv')[0])
```

```
X =  1.50E+11  +/-  1E+06
```

[10]:
```
(149597873306.8813, 1068841.3742901)
```

[11]:
```
estimation(caracteristiques_moyennes('Terre_H.csv')[1])
```

```
X =  3.15E+07  +/-  1E+05
```

[11]:
```
(31536000.0, 100594.07537226035)
```