TRACK YOUR ATMOSPHERE

Intellectual Output IO3

Étude des satellites du système GPS

GNSS = Global Navigation Satellite Systems

Erasmus+ Projekt 2017-1-DE02-KA202-004229

Version 04/20*FR*

Co-funded by the Erasmus+ Programme of the European Union

Table des matières

A. Préambule....2
B. Prérequis....2
C. La base de données Tryatdata....2
I. Structures des tables....2
II. Premier accès à la base en python....3
D. Étude d'un satellite....3
I. Enregistrements....4
II. Période d'échantillonnage....4
III. Transformée de Fourier et analyse des résultats....5
IV. Cohérence des résultats obtenus....6
1. Calculs avec la latitude....6
2. Et la rotation de la terre ?....7
3. Autre méthode....7
4. Pour aller plus loin....10

A. Préambule

Ce document propose, à l'aide des données enregistrées dans la base de données du projet TRYAT, d'analyser puis modéliser les trajectoires des satellites du système GPS.

Pour pouvoir l'utiliser, il faudra avoir installé Python3, les bibliothèques pymysql, pynum et avoir un accès à internet pour interroger la base de données.

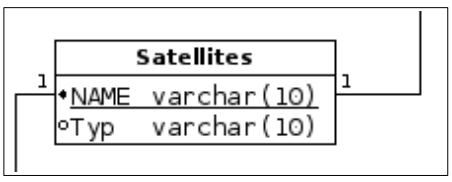
B. Prérequis

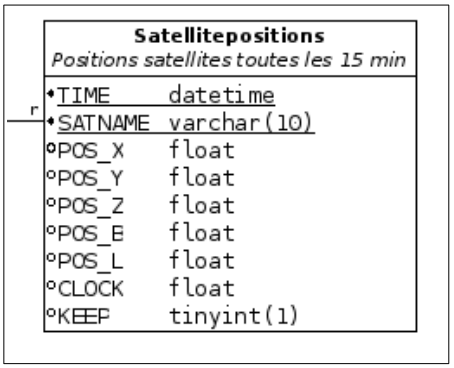
L'utilisation de ce document suppose des bases de programmation en Python, des notions de cinématique et de traitement du signal. La partie Mysql est suffisamment guidée pour être traitée par un débutant.

C. La base de données Tryatdata

I. Structures des tables

Dans cette base nous allons utiliser deux tables, la table « Satellites » qui référence les satellites de la myriade du système GPS et la table « Satellitepositions » qui donne la position du satellite à différents instants.
Leur structures sont les suivantes :





II. Premier accès à la base en python

L'identifiant est « tryatlsc », le mot de passe est « tryatpau », le serveur est « saint-cricq.com »
Bibliothèque : python3-pymysql

```
# Nous chargeons la bibliothèque ds outils mysql
import pymysql
#Nous définissons les accès en python ainsi :
mysqlhost = 'saint-cricq.com'
mysqluser = 'tryatlsc'
mysqlpass = 'tryatpau'
datenbank = 'tryatdata'
#les codes ci-dessous permettent ensuite de lancer des requêtes dans la base en tant que tryatlsc :
connexion = pymysql.connect( host=mysqlhost, user=mysqluser, passwd=mysqlpass, db=datenbank, charset='utf8' )
sql = "SELECT * FROM Satellites order by NAME ;"
cur = connexion.cursor(pymysql.cursors.DictCursor)
cur.execute( sql, ())
result = cur.fetchall() # retour dans un tuple
print(result)
connexion.commit()
connexion.close()
```

Le retour sous forme de dictionnaire stocké dans un tuple est le suivant :

```
[{'Typ': 'G', 'NAME': 'G01'}, {'Typ': 'G', 'NAME': 'G02'}, {'Typ': 'G', 'NAME': 'G03'}, {'Typ': 'G', 'NAME': 'G05'}, {'Typ': 'G', 'NAME': 'G06'}, {'Typ': 'G', 'NAME': 'G07'}, {'Typ': 'G', 'NAME': 'G08'}, {'Typ': 'G', 'NAME': 'G09'}, {'Typ': 'G', 'NAME': 'G10'}, {'Typ': 'G', 'NAME': 'G11'}, {'Typ': 'G', 'NAME': 'G12'}, {'Typ': 'G', 'NAME': 'G13'}, {'Typ': 'G', 'NAME': 'G14'}, {'Typ': 'G', 'NAME': 'G15'}, {'Typ': 'G', 'NAME': 'G16'}, {'Typ': 'G', 'NAME': 'G17'}, {'Typ': 'G', 'NAME': 'G18'}, {'Typ': 'G', 'NAME': 'G19'}, {'Typ': 'G', 'NAME': 'G20'}, {'Typ': 'G', 'NAME': 'G21'}, {'Typ': 'G', 'NAME': 'G22'}, {'Typ': 'G', 'NAME': 'G23'}, {'Typ': 'G', 'NAME': 'G24'}, {'Typ': 'G', 'NAME': 'G25'}, {'Typ': 'G', 'NAME': 'G26'}, {'Typ': 'G', 'NAME': 'G27'}, {'Typ': 'G', 'NAME': 'G28'}, {'Typ': 'G', 'NAME':
'G29'}, {'Typ': 'G', 'NAME': 'G30'}, {'Typ': 'G', 'NAME': 'G31'}, {'Typ': 'G', 'NAME': 'G32'}]
```

Pour obtenir le nombre de satellites stockés dans la base il aurait fallut modifier la requête ainsi :

```
sql = "SELECT COUNT(*) FROM Satellites ;"
```

Télécharger le programme http://saint-cricq.com:82/sources/EtudeSatellite1.py

D. Étude d'un satellite

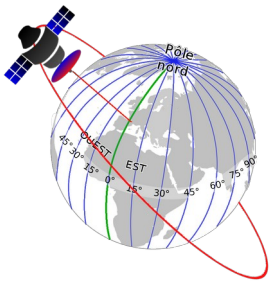
Nous allons maintenant nous intéresser au satellite nommé **G31** et demander combien de positions sont enregistrées pour ce satellite :
La requête SQL devient alors :

```
sql = """SELECT COUNT(*) FROM Satellitepositions WHERE SATNAME=  "G31" ;"""
```

On notera le triplement des guillemets pour les différencier de ceux encadrant le texte G31

Co-funded by the Erasmus+ Programme of the European Union

(échappement)
Le retour donne : [{'COUNT(*)': 5088}]. Il existe donc 5088 positions enregistrées dans la base de données pour le satellite G31.



I. Enregistrements

Grâce aux données enregistrées, nous allons maintenant essayer d'en savoir plus sur la trajectoire de ce satellite. Pour ce faire, nous allons chercher la fréquence de passage à une longitude donnée (qui pourrait par exemple être le méridien de Greenwich coloré en vert), sachant cela, et avec les lois de Kepler, nous pourrons en déduire sa période, sa vitesse et son altitude.

La requête suivante permet d'extraire toutes les positions enregistrées :

```
sql = """SELECT TIME, POS_L FROM Satellitepositions WHERE SATNAME= "G31" ORDER BY TIME;"""
```

A la date où ce programme est lancé, nous obtenons 4896 postions enregistrées

Nous allons maintenant traiter ce résultat, grâce à un algorithme de FFT pour en extraire la rotation de passage du satellite. Pour cela, nous allons utiliser de nouvelles bibliothèques python :

```
import pymysql, time
from scipy.fftpack import fft
import numpy as np
import matplotlib.pylab as pl
from math import *
```

Quelques constantes pour utiliser les lois de Kepler :

```
# Données constantes
G=6.67384E-11
Mt=5.9736E24
Rt=6371E3
```

II. Période d'échantillonnage

```
# enregistrement des positions dans deux listes (longitude, temps)
listePositions=[]
listeTemps=[]
for element in dictPositions :
        listePositions.append(element["POS_L"])
        listeTemps.append(element["TIME"])
print("Fin de remplissage des listes de positions/dates")
Te=listeTemps[1]-listeTemps[0]
```

```
print("Période d'échantillonnnage au format date : "+str(Te))
print("Période d'échantillonnnage au format secondes : "+str(Te.seconds))
Fe=1/Te.seconds
```

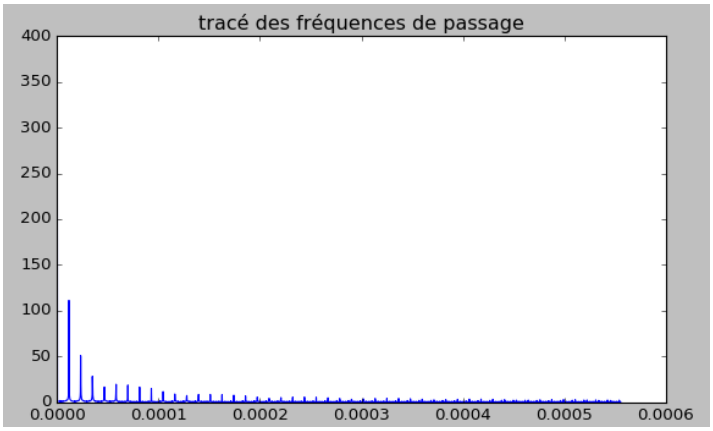
La période obtenue est de 900s, soit 15 min, nous connaissons donc la position du satellite G31 toutes les 15 min. Pour être plus précis, il aurait fallu faire une période moyenne sur tous les échantillonnages, mais dans ce cas ce n'était pas nécessaire.

III. Transformée de Fourier et analyse des résultats

La partie de programme ci-dessous calcule la FFT de la trajectoire du satellite et en trace le spectre fréquentiel :

```
xFFT = np.linspace(0.0,Fe/2, nbrValeurs/2)
# on trace la FFT
pl.plot(xFFT, 2.0/nbrValeurs * abs(posFFT[:nbrValeurs//2]))  # affichage
pl.title("tracé des fréquences de passage")
pl.show()
```

Nous obtenons le résultat suivant :



Nous remarquons une raie principale dont nous cherchons la fréquence en parcourant le tableau résultant de la FFT, nous en profitons pour stocker les valeurs dans un fichier CSV afin de pouvoir les ré-utiliser avec d'autres logiciels.

```
fichier=open("data.csv", "w")
# Détection du maximum
for i in range (1,int(nbrValeurs/2)) :
        #print(abs(posFFT[i]),xFFT[i])
        fichier.write(str(abs(posFFT[i]))+";"+str(xFFT[i])+"\n")
        if((abs(posFFT[i])) > lonMax) :
                lonMax=abs(posFFT[i])
                fMax=xFFT[i]
                iMax=i
                #print("Maximum\n")
fichier.close()
```

À l'aide de la 3ème loi de Kepler nous calculons les paramètres de la trajectoire :

$$\frac{T^2}{a^3}=\frac{4.\pi^{2.}\lambda}{G.M_{Terre}}$$
$$G=6{,}67384^{-11}\,SI$$
$$\lambda=1+\frac{M_{Satellite}}{M_{Terre}}$$

```
Tsatellite=1/fMax
print("Période : "+str(Tsatellite)+" s soit "+str(Tsatellite/3600) +" h")
rayon=(Tsatellite**(2)*G*Mt/(4*pi**2))**(1/3)
print("Le rayon de l'orbite de l'ISS est : "+str(rayon))
print("Son altitude moyenne est : "+str(rayon-Rt)+" m")
```


Co-funded by the Erasmus+ Programme of the European Union

```
print("Sa vitesse est de : "+ str(2*pi*rayon/Tsatellite) +" m/s = "+str(2*pi*rayon/(Tsatellite)*3.6)+" km/h")
```

Les résultats sont les suivants :

- *Il y a 4896 positions enregistrées*
- *Fin de remplissage des listes de positions/dates*
- *Période d'échantillonnage au format date : 0:15:00*
- *Période d'échantillonnage au format secondes : 900*

- *Amplitude maximum : 272110.221754*
- *272110.221754 1.15788039777e-05 51*
- *Période : 86364.7058824 s soit **23.9901960784 h***
- *Le rayon de l'orbite de l'ISS est : 42231994.8759*
- *Son altitude moyenne est : **35860994.8759 m***
- *Sa vitesse est de : 3072.45242123 m/s = 11060.8287164 km/h*

Code complet :
saint-cricq.com:82/sources/EtudeSatelliteFFT.py

IV. Cohérence des résultats obtenus

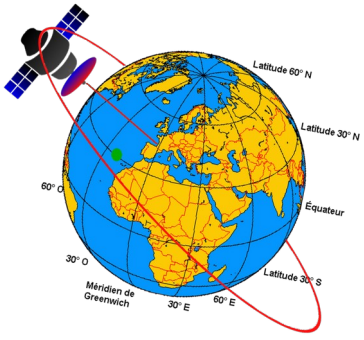
Un petit tour sur la Wikipedia nous donne l'information suivante : *« Le GPS comprend au moins 24 satellites orbitant à 20 200 km d'altitude. ... »*, et pourtant le programme nous annonce une altitude de 35 860 994.8759 m soit 36 000 km, ce qui correspond à l'altitude des satellites géo-stationnaires, ce qui est confirmé par la période de l'orbite qui vaut à peu près 24h00 (23.9901960784 h).

Les résultats obtenus sont donc faux. Nous allons donc chercher d'où vient l'erreur.

1. Calculs avec la latitude

Nous allons calculer la période de passage par une même latitude, pour cela, il suffit dans le programme de changer POS_L par POS_B, nous obtenons les résultats suivants :

- *Période d'échantillonnage au format date : 0:15:00*
- *Période d'échantillonnage au format secondes : 900*
- *Amplitude maximum : 112431.610303*
- *112431.610303 2.31576079553e-05 102*
- *Période : 43182.3529412 s soit **11.9950980392 h***
- *Le rayon de l'orbite de l'ISS est : 26604489.6616*
- *Son altitude moyenne est : **20233489.6616 m***
- *Sa vitesse est de : 3871.04748031 m/s = 13935.7709291 km/h*



Avons nous pour autant résolu le problème ? Si nous regardons l'image à droite, nous voyons que le satellite passera par la même latitude(notée avec le point vert sur le tropique du Cancer) deux fois au cours d'une révolution, de part et d'autre du globe terrestre. Il est normal de trouver une fréquence de passage par une même latitude deux fois plus grande que celle par une même longitude. Nous retombons par conséquent sur les mêmes résultats.

2. Et la rotation de la terre ?

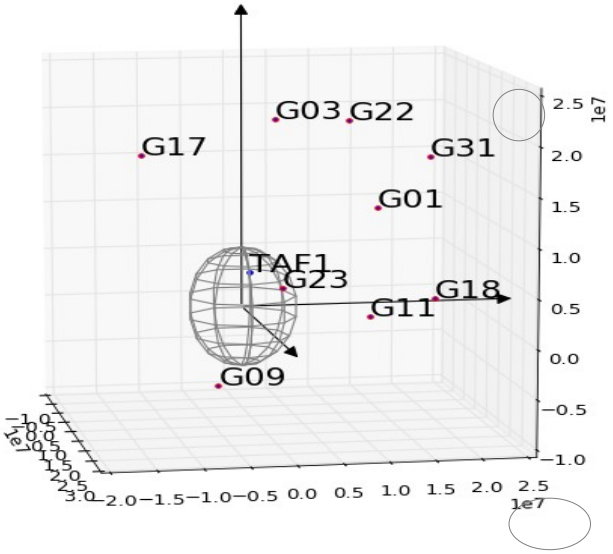
Le positionnement du satellite est relevé par rapport à un point sur la terre et non par rapport au centre de la terre. Ce point se déplace au rythme d'un tour toutes les 24h00, lorsque le satellite a fait un tour de la terre, celle-ci a aussi tourné, et si l'altitude du satellite est de 20 000 km, il tourne deux fois plus vite que la terre. Quand il aura parcouru un tour, la terre en aura fait 1/2, c'est donc à son deuxième

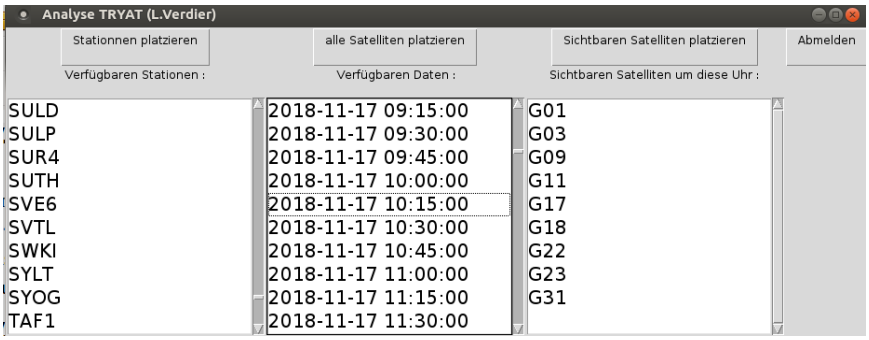
tour que le satellite reverra la même longitude.

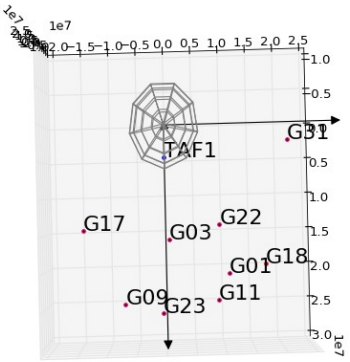
3. Autre méthode

Il est donc nécessaire de calculer les paramètres de la trajectoire du satellite G31 en utilisant un autre procédé.

Nous allons utiliser un programme qui représente la position des satellites en trois dimensions, le 17/11/2018 à 10:15:00, ce programme nous indique que G31 était visible de la station TAF1, située à Pau à proximité du méridien de Greenwich, en observant à partir du pôle nord (seconde image), on peut même voir qu'il était en limite de visibilité :







Nous allons donc utiliser la représentation 3D pour tracer la trajectoire de G31 sur une journée.

La requête dans la base de données est donc :

```
SELECT * FROM Satellitepositions WHERE TIME >= '2018-11-17 10:15:00' AND TIME <='2018-11-18 10:15:00' AND SATNAME LIKE 'G31';
```

Nous allons intégrer une nouvelle bibliothèque et quelques autres outils :

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
```

Nous ajoutons aussi un objet pour tracer les vecteurs de repérage dans l'espace
class Arrow3D(FancyArrowPatch):

```
class Arrow3D(FancyArrowPatch):
def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0, 0), (0, 0), *args, **kwargs)
        self._verts3d = xs, ys, zs
```


Co-funded by the Erasmus+ Programme of the European Union

```
def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0], ys[0]), (xs[1], ys[1]))
        FancyArrowPatch.draw(self, renderer)
```

et le programme principal :

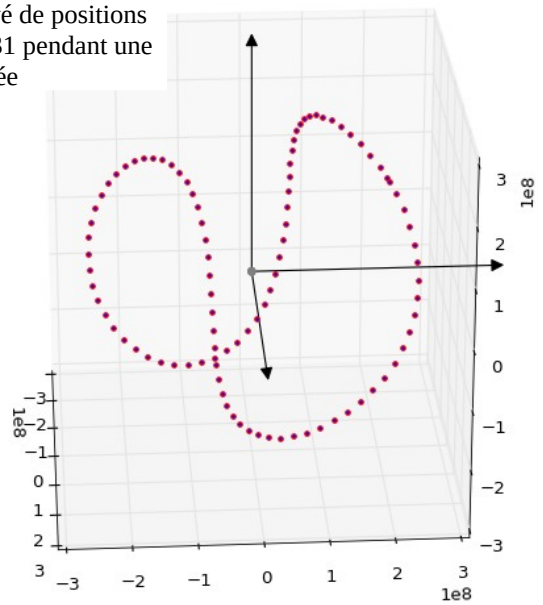
```
# ----------------  Tracé des trajectoires en 3D
sql = """SELECT * FROM Satellitepositions WHERE TIME >= "2018-11-17 10:15:00" AND TIME <="2018-11-18 10:15:00" AND SATNAME LIKE "G31" ;"""
dictPositions3d={} # dictionnaire contenant les positions du satellite stockées dans la base de données
connexion = pymysql.connect( host=mysqlhost, user=mysqluser, passwd=mysqlpass, db=datenbank, charset='utf8' )
cur = connexion.cursor(pymysql.cursors.DictCursor)
cur.execute( sql, ())
dictPositions3d = cur.fetchall() # retour dans unn tuple
#print(dictPositions)
connexion.commit()
connexion.close()

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_aspect("equal")

# dessin de la terre
u, v = np.mgrid[0:2*np.pi:10j, 0:np.pi:10j]# remplit un tableau 2D pour le maillage
x = 0.90*Rt*np.cos(u)*np.sin(v)
y = 0.90*Rt*np.sin(u)*np.sin(v)
z = 0.90*Rt*np.cos(v)
ax.plot_wireframe(x, y, z, color="grey")

lfleche= 4E8
# dessin du repère avec comme axe OX la direction de Greenwich
ox = Arrow3D([0, lfleche], [0, 0], [0, 0], mutation_scale=20, lw=1, arrowstyle="-|>", color="k")
ax.add_artist(ox)
oy = Arrow3D([0, 0], [0, lfleche], [0, 0], mutation_scale=20, lw=1, arrowstyle="-|>", color="k")
ax.add_artist(oy)
oz = Arrow3D([0, 0], [0, 0], [0, lfleche], mutation_scale=20, lw=1, arrowstyle="-|>", color="k")
ax.add_artist(oz)
# placer les stations scatter : confetti
for element in dictPositions3d:
    ax.scatter(int(element["POS_X"])*10E3,  int(element["POS_Y"])*10E3,  int(element["POS_Z"])*10E3,  color="r", s=10)
plt.show()
```

Code complet :
saint-cricq.com:82/sources/EtudeSatelliteFFT3D.py

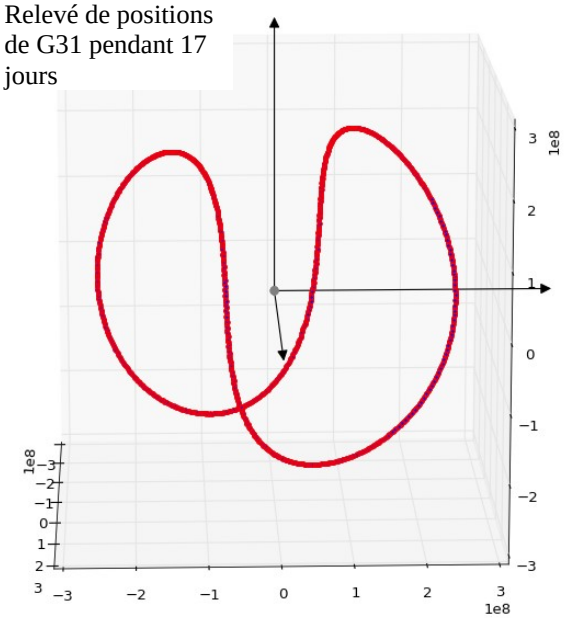
Qui renvoie la trajectoire de G31 ci-dessous, par rapport à l'observateur placé sur la terre

Pour vérifier si la période est une demi période terrestre nous pouvons modifier la plage de positions de G31 en prenant sur plusieurs jours (17). On constate qu'il n'y a pas de glissement en longitude, la période de G31 est donc exactement la moitié de celle de la Terre.

Relevé de positions de G31 pendant une journée



Relevé de positions de G31 pendant 17 jours



4. Pour aller plus loin

- A partir des mesures, modifier le programme pour donner l'angle que fait le plan de la trajectoire de G31 avec le plan passant par l'équateur.
- Calculer l'altitude de G31
- En prenant les positions des satellites vus par TAF1 à une date donnée, calculer les différences de temps entre l'émission et la réception des trames GPS.

Co-funded by the Erasmus+ Programme of the European Union