TP INFORMATIQUE
N°2
CORRIGE

Question 1: L'unité de l'accélération fournie par l'accéléromètre du téléphone portable est le g (g=9.81m.s^{-2}). Modifier le programme afin d'obtenir l'accélération en m.s^{-2}.

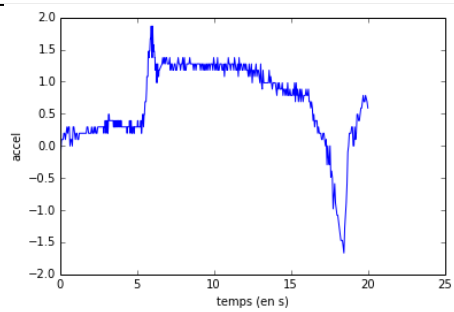
Il suffit de modifier la ligne créant la liste accélération:

```
acceleration.append(9.81*float(ligne_data[5]))
```

Question 2: Afficher la courbe de l'accélération longitudinale.

```
plt.plot(temps,acceleration)
plt.xlabel("temps (en s)")
plt.ylabel("accel (en m/s2)")
```

On obtient la courbe suivante:



Question 3: Afficher l'accélération latérale (selon x). En déduire si le tramway est vraiment en ligne droite. Remarque: la précision de mesure du capteur est de 0.05 m/s^2.

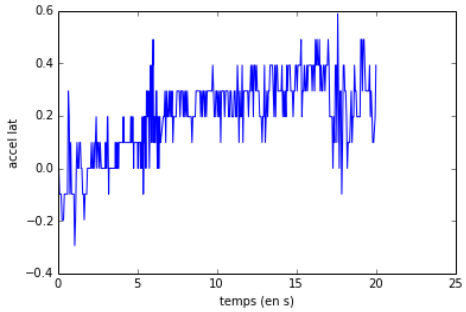
On crée la liste accel_lat (en utilisant la même boucle que pour la liste précédente):

```
for i in range(n-2):
        .............
        .............
        accel_lat.append(9.81*float(ligne_data[3]))
```

Pour le tracé de la courbe:

```
plt.figure()  #nouvelle figure
plt.plot(temps,accel_lat)
plt.xlabel("temps (en s)")
plt.ylabel("accel lat (en m/s2)")
```

On obtient:



On observe de petites accélérations ne dépassant pas 0,6m/s^2. Cela est dû aux vibrations du tramway qui peuvent être causées par les imperfections du guidage des rails, du vent, de la précision du capteur. On peut néanmoins considérer qu'il s'agit bien d'une ligne droite.

Question 4: A l'aide de la fonction **max** définie dans Python afficher l'accélération longitudinale et latérale maximum.

```
print(max(acceleration))
print(max(accel_lat))
```

On obtient 1,86 m/s^2 pour l'accélération longitudinale et 0,59 m/s^2 pour l'accélération latérale.

Question 5: Ecrire une fonction integr1(f,t) utilisant la méthode des rectangles, ayant pour argument d'entrée deux listes f et t, et retournant une liste de valeurs prises par F en chacun des instants de la liste t telle que:

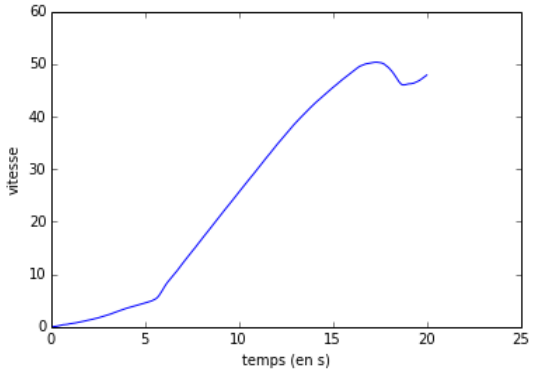
$$\begin{cases} F(t) = \int_{t_0}^{t} f(u).du \\ F(t_0) = 0 \end{cases}$$

```
def integr1(f,t):
    F=[0]
    n=len(t)
    for k in range(1,n):
        F.append(F[k-1]+(t[k]-t[k-1])*f[k-1])
    return (F)
```

Utiliser cette fonction afin d'obtenir la vitesse du tramway en km/h sous forme d'une liste qu'on nommera **vitesse_rec**. Afficher la courbe de la vitesse.

```
vitesse_rec=integr1(acceleration,temps)
vitesse_rec=[3.6*v for v in vitesse_rec]
#Tracé de la courbe de vitesse
plt.figure()
plt.plot(temps,vitesse_rec)
plt.xlabel("temps (en s)")
plt.ylabel("vitesse (en m/s)")
```

On obtient:



Question 6: A l'aide de la fonction max de python, déterminer la valeur maximum de la vitesse. Le conducteur respecte-t-il la réglementation du BIRMTG ?

On saisit:

```
print(max(vitesse_rec))
```

On obtient la valeur de la vitesse maxi, 50,35 km/h

Question 7: Ecrire une fonction integr2(f,t) utilisant la méthode des trapèzes, ayant pour argument d'entrée deux listes f et t, et retournant une liste de valeurs prises par F en chacun des instants de la liste t telle que:

$$\begin{cases} F(t) = \int_{t_0}^{t} f(u).du \\ F(t_0) = 0 \end{cases}$$

Utiliser cette fonction afin d'obtenir la liste de vitesse avec la méthode des trapèzes (**vitesse_trap**).

```
def integr2(f,t):
    F=[0]
    n=len(t)
    for k in range(1,n):
        F.append(F[k-1]+(t[k]-t[k-1])*(f[k]+f[k-1])/2)
    return (F)

vitesse_trap=integr2(acceleration,temps)
vitesse_trap=[3.6*v for v in vitesse_trap]
#Tracé de la courbe de vitesse
plt.figure()
plt.plot(temps,vitesse_trap)
plt.xlabel("temps (en s)")
plt.ylabel("vitesse (en m/s)")
```

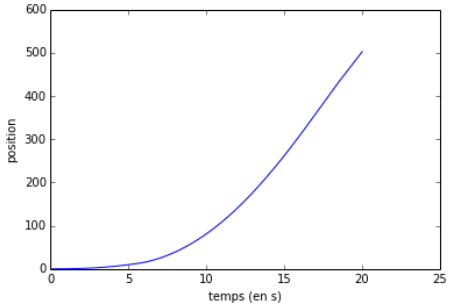
On obtient la même courbe qu'avec la méthode des rectangles (les valeurs étant plus précises).

Question 8: Utiliser une des fonctions précédentes afin d'obtenir la position du tramway (en m) sous forme d'une liste. Afficher la courbe de la position. Quelle est la distance parcourue ?

On utilise la fonction integr1 pour trouver la position à partir de la vitesse.

```
position=integr1(vitesse_rec,temps)
plt.figure()
plt.plot(temps,position)
plt.xlabel("temps (en s)")
plt.ylabel("position (en m)")
plt.show()
```

On obtient la courbe suivante:



En saisissant position[-1], on obtient la dernière valeur de la liste qui est la distance totale parcourue, à savoir 502,4 m.

Question 9: Ecrire une fonction **filtre_mg**(L,N) qui prend en argument d'entrée une liste L ainsi que l'ordre de la moyenne glissante N et qui retourne la liste L filtrée.

```
def filtre_mg(L,N):
    liste_mg=[]
    for p in range(N-1,len(L)):
        mg_k=0
        for k in range(N):
            mg_k=mg_k+L[p-k]
        liste_mg=liste_mg+[mg_k/N]
    return liste_mg
```

Autre façon de définir la fonction, à partir de la deuxième formule proposée:

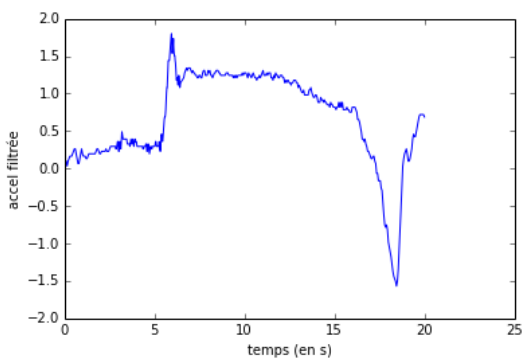
```
def filtre_mg(L,N):
        liste_mg=[]
        mg_k=0
        for k in range(N-1):
                mg_k=+mg_k/N
        for i in range(N-1,len(L)):
                 mg_k=mg_k+(L[i]-L[i-N])/N
                 liste_mg.append(mg_k)
        return (liste_mg)
```

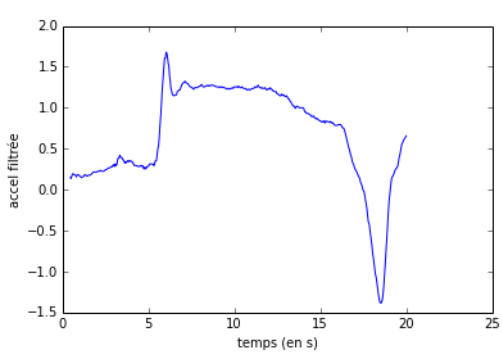
Question 10: Utiliser cette fonction afin de tracer la courbe de l'accélération filtrée, on pourra tester différentes valeurs de N afin d'avoir un filtrage acceptable.

```
acceleration_filtre3=filtre_mg(acceleration,3)
acceleration_filtre10=filtre_mg(acceleration,10)

plt.figure()
plt.plot(temps[2:],acceleration_filtre3)
plt.xlabel("temps (en s)")
plt.ylabel("accel filtrée (en m/s^2)")
plt.figure()
plt.plot(temps[9:],acceleration_filtre10)
plt.xlabel("temps (en s)")
plt.ylabel("accel filtrée(en m/s^2)")
```

On obtient les courbes suivantes:





On remarque bien entendu que plus la valeur de N est grande, plus la courbe est lissée, en revanche on ne peut pas prendre un ordre trop grand sinon on perd beaucoup de valeurs.

Question 11: En utilisant le schéma d'Euler explicite, donner la relation de récurrence définissant s_i en fonction de e_{i-1}, s_{i-1}, t_i, t_{i-1} et T pour $i \in [\![1, n-1]\!]$.

$$s_i = s_{i-1} + \frac{(t_i - t_{i-1}).(e_{i-1} - s_{i-1})}{T}$$

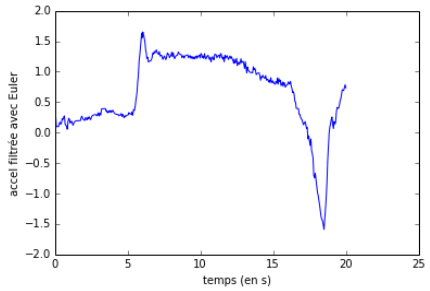
Question 12: Définir une fonction Euler(t, e) ayant pour argument d'entrée deux listes de valeur t et e de même dimension renvoyant la liste des valeurs approchées de s associées aux valeurs de la liste e.

```
def Euler(t,e):
    s=[e[0]]
    T=0.1
    for k in range (1,len(e)):
        s.append(s[k-1]+(t[k]-t[k-1])*(e[k-1]-s[k-1])/T)
    return (s)
```

Question 13: Utiliser la fonction précédente afin de tracer la courbe d'accélération filtrée.

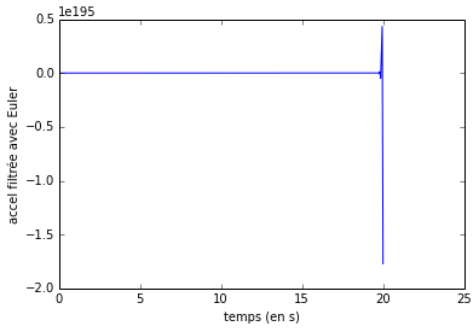
```
a_filtre=Euler(temps,acceleration)
plt.figure()
plt.plot(temps,a_filtre)
plt.xlabel("temps (en s)")
plt.ylabel("accel filtrée avec Euler (en m/s$^2$)")
```

On obtient:



Le filtrage est relativement satisfaisant. La difficulté est de trouver une valeur acceptable pour la constante de temps T, ce qui montre les limites du filtrage basse fréquence ainsi que les limites de la méthode d'Euler. En effet, si T est trop grand, on filtre trop de fréquences et on perd le signal réel, si T est trop faible on ne filtre pas suffisamment le bruit. D'autre part on se rend compte que pour une valeur de T trop faible, la méthode diverge car T devient alors inférieur à la résolution de l'acquisition.

Pour T=0,01s, on obtient:



Pour T=0,5s, on obtient: