INFORMATIQUE

Les dictionnaires en Python

Gestion de fichiers lecture / écriture de données :

La plupart des appareils électroniques permettent d'enregistrer des données dans un fichier :

- Ordinateur [Tableur, Scilab, Synchronie, Python]
- Base de données en ligne
- Oscilloscope en TP.
- Smartphone, [ex : accéléromètre, boussole]

Différents formats sont possibles pour encoder les données :

- Fichier binaire (voire compressé) pour les grands volumes de données.

Les données d'une base de données sont gardées sur des serveurs.
Une donnée est désignée par un numéro d'armoire, de disque, de cylindre, de face, de secteur, de fichier etc ... ... Il faut la trouver puis la décompresser.

- Fichier texte de type CSV [Comma Seperated Values]

Les données sont écrites de manière textuelle ligne par ligne.
Les différentes informations sont écrites dans des «champs» séparés par un «séparateur»

Différents séparateurs types :	,	;	|	espace(s)	tabulation

Rq : une ligne se termine toujours par un caractère de retour à la ligne [ASCII 13 soit «**\n**» en Python]
=> ne pas confondre la ligne et son affichage qui dépend de la taille de l'écran.

On n'utilisera pas le point «.» dot comme séparateur, car il désigne la virgule en notation scientifique utilisée en informatique [anglophone].

Structure d'un fichier de donnée CSV :

Il n'y a pas de format imposé mais en général, on a toujours une première ligne appellée entête, [car elle vient en tête de fichier] qui indique ce que contiennent les champs

Ensuite chaque ligne reproduit la succession des champs indiqués dans l'entête, chacun avec le type et selon le format qui lui est propre (entier, réel, chaîne de caractères, date etc...).
Tous les champs sont séparés par un même séparateur.

Exemple : Données d'une trajectoire dans un fichier CSV écrit en Python.

```
i, time, x[i], y[i], vx[i], vy[i], ax, ay
    1,  1.9980e-03,  1.9977e-02,  1.9977e-02,  9.4335e+00,  9.4139e+00,  -2.8276e+02,  -2.9257e+02
    2,  3.9960e-03,  3.8825e-02,  3.8786e-02,  8.9312e+00,  8.8930e+00,  -2.5145e+02,  -2.6073e+02
    3,  5.9940e-03,  5.6670e-02,  5.6554e-02,  8.4813e+00,  8.4255e+00,  -2.2513e+02,  -2.3398e+02
    4,  7.9920e-03,  7.3615e-02,  7.3388e-02,  8.0762e+00,  8.0034e+00,  -2.0279e+02,  -2.1126e+02
    5,  9.9900e-03,  8.9752e-02,  8.9379e-02,  7.7092e+00,  7.6202e+00,  -1.8365e+02,  -1.9181e+02
                                                ..............
 1000,  1.9980e+00,  1.1717e+00, -2.9377e+00,  9.3099e-04, -2.2147e+00,  -4.1606e-03,  -1.8411e-05
```

La mise en forme de votre fichier est essentielle car la récupération des données en dépend.

Syntaxe de base en Python :

On utilise la commande **open** qui renvoie un objet de type **fichier**
Il existe toutes sortes de méthodes qui agissent sur les fichiers pour lire et écrire notamment :

```
myfile='fichier.csv'

fichier=open(myfile,'r')     #ouvre le fichier pour lecture seule  ['r+' pour lecture/écriture]
fichier.close()              #ferme le fichier

liste=fichier.readlines()    #crée une liste de chaine : chaque élément est une ligne texte du fichier
fichier.write()              #écrit dans le fichier à partir de la position du curseur fichier.
```

Les options d'ouverture :

'w' : écriture seule	ATTENTION : les options 'w' effacent tout le fichier avant d'ouvrir
'w+' : lecture/écriture	
'a' : ajout	Ajout ouvre le fichier et place le cuseur en fin de fichier, pour ajouter des données sans effacer le contenu

Formats de base :

- {:5d} -> entier sur 5 digits minimum
- {:10.4e} -> notation scientifique : 4 chiffres après la virgule 10 caractères tout compris dont le signe.
- {:10.4f} -> notation flottants : idem sans l'exposant

```
print("{:15d}".format(12345678))      | '       12345678'

print("{:10.2e}".format(1234.5678))   | '   1.23e+03'
print("{:10.2e}".format(-1234.5678))  | '  -1.23e+03'

print("{:4.2f}".format(-1234.5678))   | '-1234.57'
print("{:4.2f}".format(1234.5678))    | '1234.57'
```

Rq :

On voit que 4 est un minimum, il faut donc s'assurer que les nombres du fichier n'excèderont pas la longueur de chaine fixée.

Exemple : calcul et écriture d'une trajectoire par la méthode d'Euler

```
myfile='/Users/jcourtin/Desktop/Python/ENV/Pyzo/Workspace_PROJET/Workspace_Euler/data.csv'
fichier=open(myfile,'w')
fichier.write("{}, {}, {}, {}, {}, {}, {}, {}\n".format('i','time','x[i]','y[i]','vx[i]','vy[i]','ax','ay'))
                        #Création de l'entête
for i in range(1,N): #Boucle sur les pas de temps
    time+= .....
    .......    # (Algorithme d'intégration d'Euler non détaillé)
    .......
            #Mise en forme des résultats à enregistrer
    line=" {:5d}, {:10.4e}, {:10.4e}, {:10.4e}, {:10.4e}, {:10.4e}, {:10.4e}, {:10.4e}\n"
          .format(i,time[i],x[i],y[i],vx[i],vy[i], ax, ay)

    fichier.write(line) #Ecriture de la ligne dans le fichier

fichier.close()
```

Exemple : récupération des données de l'accéléromètre d'un iPhone

Une application iPhone envoie par email un fichier .csv d'une séquence de mesure :

```
#mieux vaut indiquer dans une variable le chemin complet du fichier depuis la racine

myfile = '/Users/jcourtin/Desktop/Python/ENV/Pyzo/Workspace_PROJET/Workspace_accelerometer/Pendule_106.csv'

fichier = open(myfile,'r')
liste = fichier.readlines()  #crée une liste de chaines contenant chacune une ligne du fichier.
fichier.close()
```

```
>>> type(fichier)
<class '_io.TextIOWrapper'>

>>> liste[0]
'time,timestamp,recordtime,accelerationX,accelerationY,accelerationZ,HeadingX,HeadingY,Headi
ngZ,TrueHeading,MagneticHeading,HeadingAccuracy,en0,pdp_ip0,DeviceOrientation,Event\n'

>>> liste[1];liste[2]                                                    boussole ...
'2014-02-02 16:09:22 +0000,1,  0.05,  0.5437164, 0.8057404, 0.02705383,  0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,2,5\n'
'2014-02-02 16:09:22 +0000,2,  0.1  ,  0.5437164, 0.8057404, 0.02705383, 0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,2,5\n'
      date ...
```

temps ax ay az

Les retours à la ligne «\n»

Il nous faut **extraire l'information** d'intérêt de chaque ligne :

Opération cruciale qui dépend de la structure du fichier !
=> bien lire le fichier

La méthode **chaine.split(',')** renvoie la liste des valeurs séparées par «,» => CSV

```
myfile='/Users/jcourtin/ .....  /ENV/Pyzo/Workspace_PROJET/Workspace_accelerometer/Pendule_106.csv'

fichier=open(myfile,'r')                    #Ouverture fichier

entete=fichier.readline()                   #Lit l'entete, et passe à la ligne


time=[];  ax=[]; ay=[]; az=[]

for L in fichier:                           #L'objet fichier est itérable ligne par ligne !!!!


    Ligne=L.split(",")                      #découpage : à l'aide du séparateur

    time += [float(Ligne[2])]
    ax +=    [float(Ligne[3])]              # Reconstruction
    ay +=    [float(Ligne[4])]
    az +=    [float(Ligne[5])]

fichier.close()                             #Fermeture fichier ...
```

Il nous faut **extraire l'information** d'intérêt de chaque ligne :

Opération cruciale qui dépend de la structure du fichier !

La méthode **chaine.index(',')** renvoie la position du prochain séparateur «,»

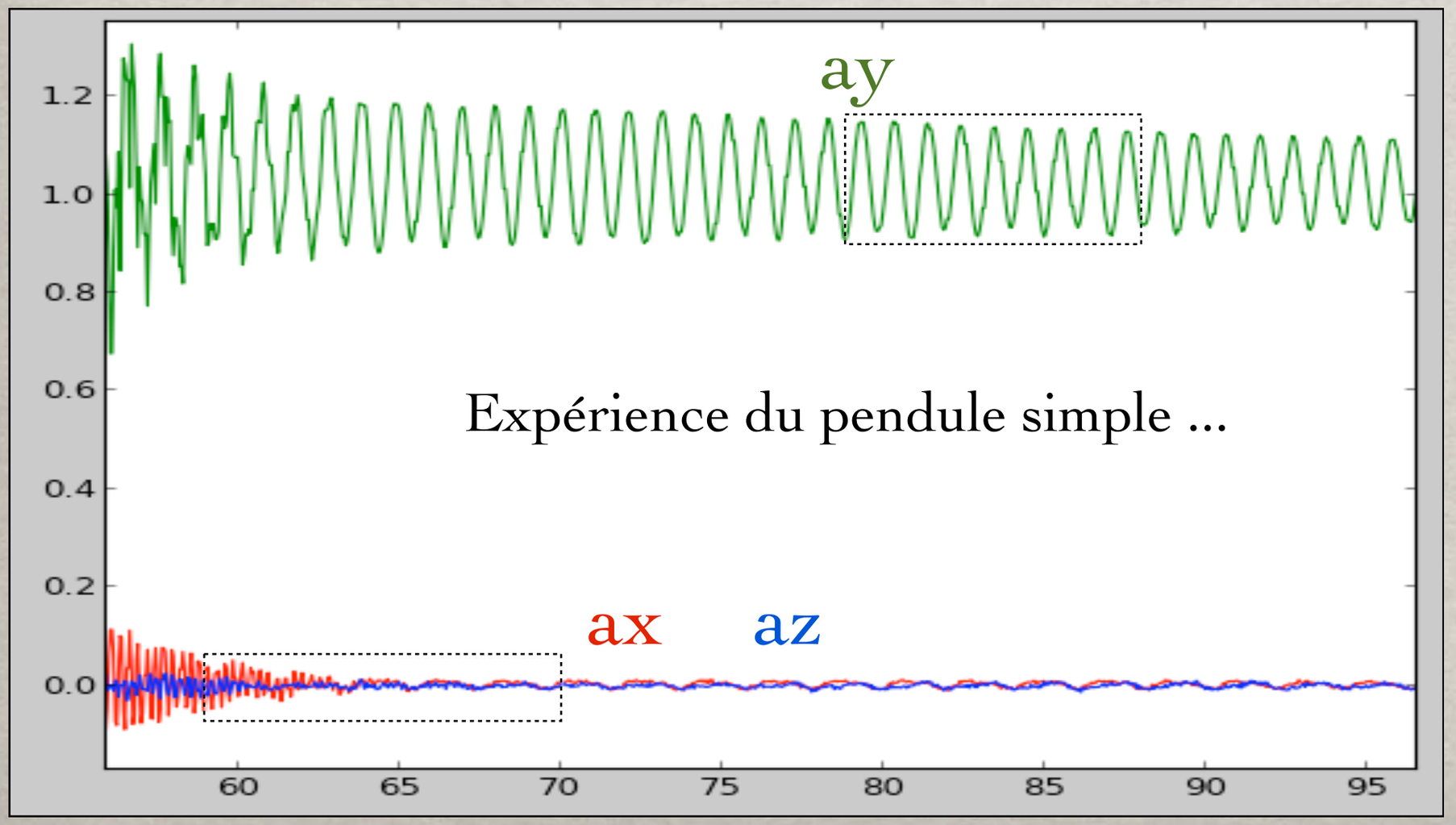
Version piéton : sans «SPLIT»

```
time=[]
ax=[]
ay=[]
az=[]
for i in range(1,len(liste)-1):
    chaine=liste[i]                          #copie la ligne dans chaine
    chaine=chaine[chaine.index(',')+1:]
    chaine=chaine[chaine.index(',')+1:]      #enlève les deux premiers champs qui sont inutiles

    time+=[float(chaine[:chaine.index(',')])] #concatène le champ suivant dans time format FLOAT
    chaine=chaine[chaine.index(',')+1:]      #enlève le champ

    ax+=[float(chaine[:chaine.index(',')])]  #concatène le champ suivant dans ax format FLOAT
    chaine=chaine[chaine.index(',')+1:]      #enlève le champ

    ay+=[float(chaine[:chaine.index(',')])]  #concatène le champ suivant dans ay format FLOAT
    chaine=chaine[chaine.index(',')+1:]      #enlève le champ

    az+=[float(chaine[:chaine.index(',')])]  #concatène le champ suivant dans az format FLOAT
    chaine=chaine[chaine.index(',')+1:]      #enlève le champ
```

time,timestamp,recordtime,accelerationX,accelerationY,accelerationZ,HeadingX,HeadingY,HeadingZ,TrueHeading,
MagneticHeading,HeadingAccuracy,en0,pdp_ip0,DeviceOrientation,Event

2014-01-31 10:16:53 +0000, 1, 0.1, -0.2920074, 0.4899902, -0.8274078, 0,0,0,0,0,-1,0.0.0.0,xx.xx.xx.xx,4,0
2014-01-31 10:16:53 +0000, 2, 0.2, -0.4313965, 0.09684753, -0.8518219, 0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,4,0
2014-01-31 10:16:53 +0000, 3, 0.3, -0.4515228, 0.02388, -1.013245, 0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,4,0
2014-01-31 10:16:53 +0000, 4, 0.4, -0.7271576, -0.5062561, -0.5905914, 0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,4,0
2014-01-31 10:16:53 +0000, 5, 0.5, -1.187729, -0.1874237, -0.1728668, 0,0,0,0,0,-1,0.0.0.0, xx.xx.xx.xx,4,0

temps ax ay az

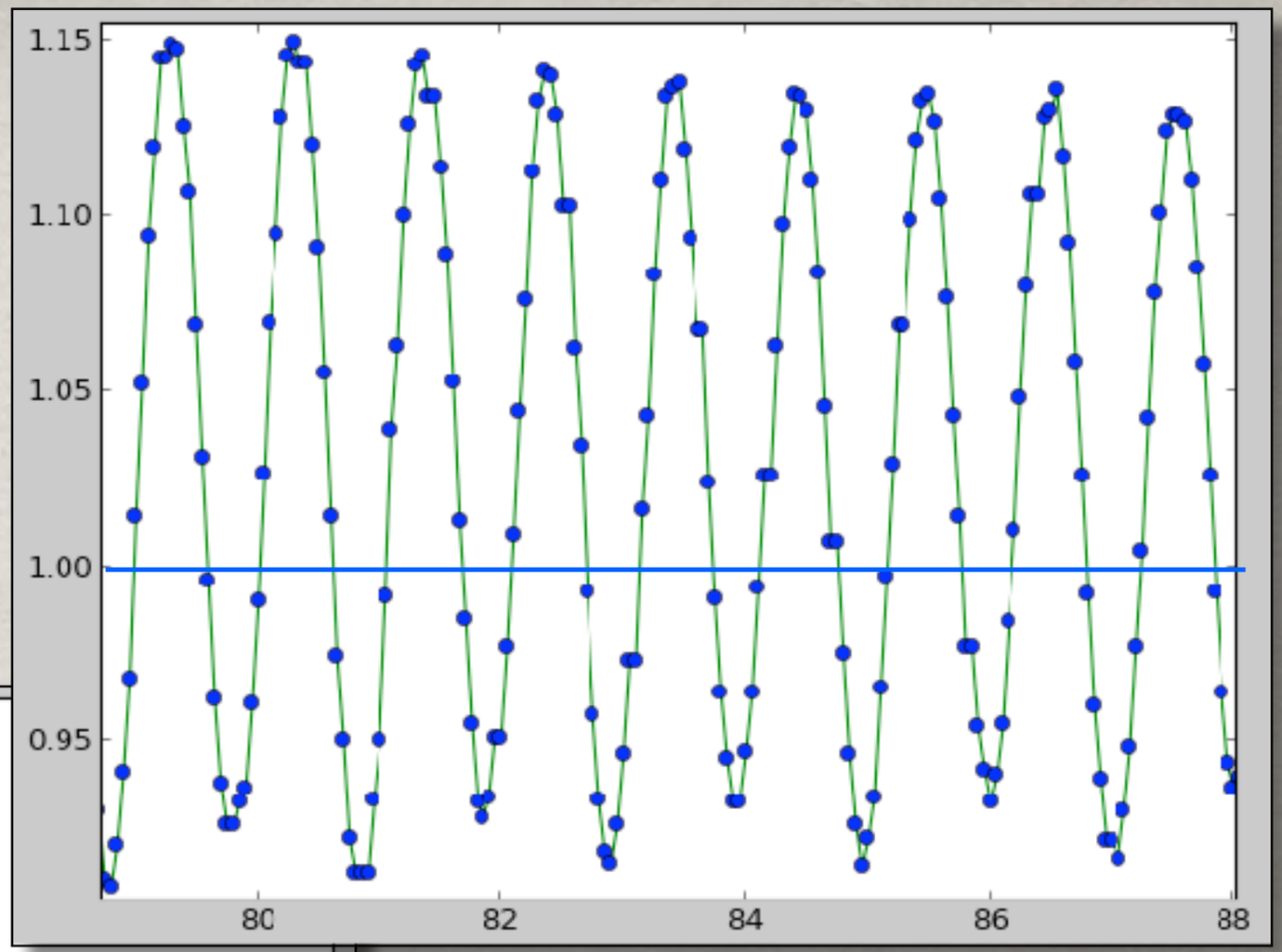


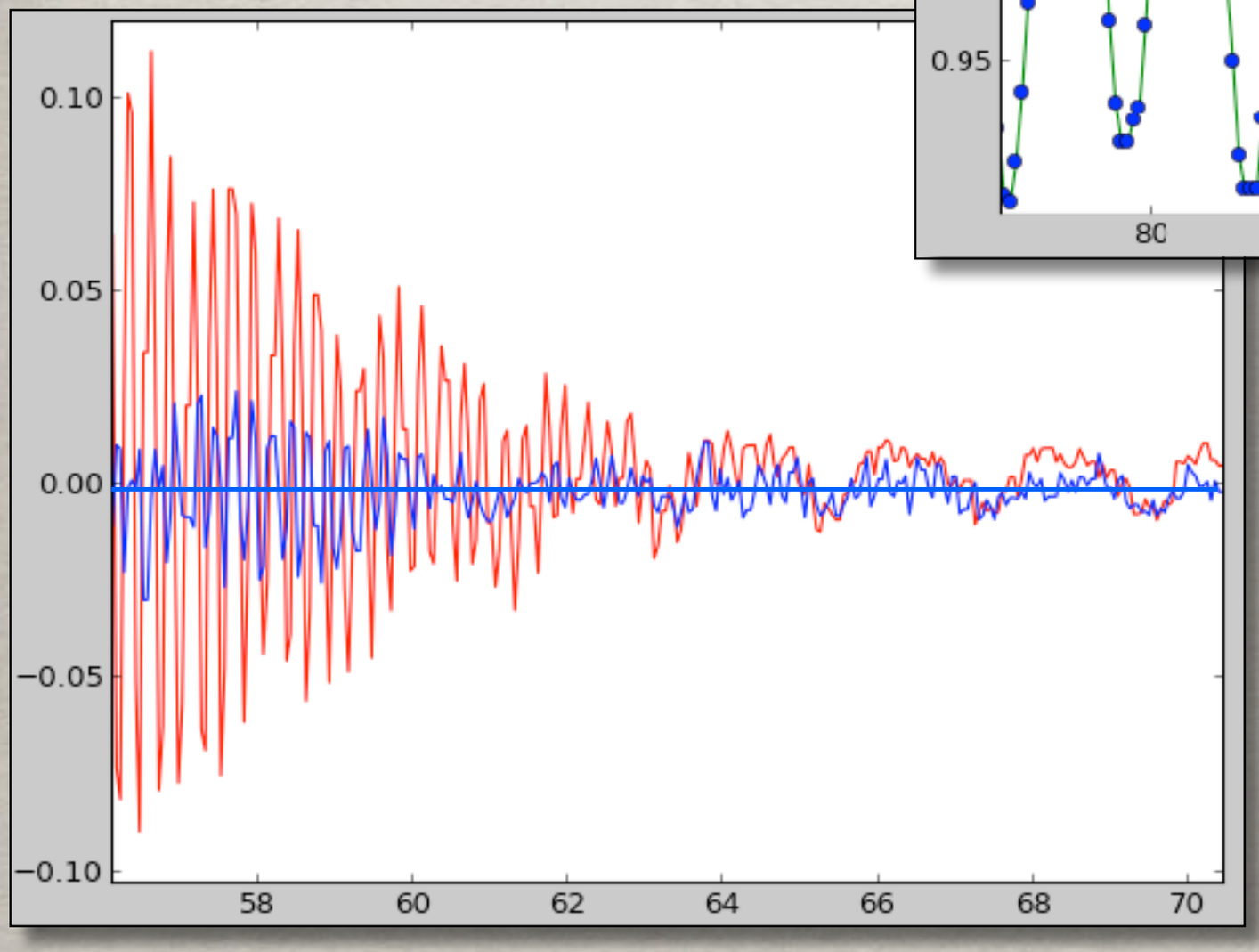
Tracé des données :

(Avec matplotlib.pyplot)

```
plt.close()
plt.plot(time,ax,'red')
plt.plot(time,ay,'green')
plt.plot(time,az,'blue')
plt.show()
```

Ay : Accélération radiale ~ g





Ax : Accélération orthoradiale

Az : Accélération longitudinale

Les signaux sont noyés dans le bruit
(de l'ordre de 10mg)

Pourquoi n'obtient-on pas $e^{-\lambda t}[A\cos(\omega t)+B\sin(\omega t)]$?

Etude d'un pendule avec l'oscilloscope :

L'angle est mesuré avec un potentiomètre
La tension mesurée par l'oscilloscope est ensuite numérisée [CAN].

On voit que le signal est «étagé» ce qui ne facilite pas les calculs de dérivé ...

Autre problème ....