# M19. Python pour la Physique et les TIPE.

## Importation des bibliothèques au début des programmes :

```
import os          # si nécessaire,  interface avec le système d'exploitation
import numpy as np                  #pour certains calculs voir module cmath
import matplotlib.pyplot as plt
```

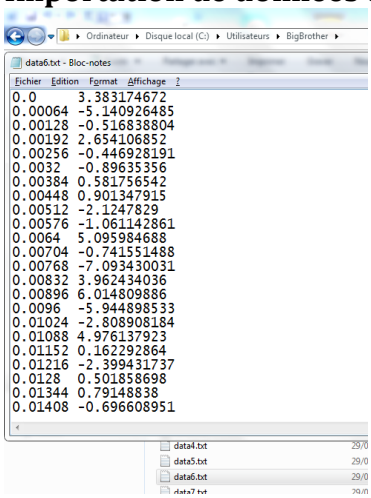
## Quel est le repertoire de travail ? :

```
rep=os.getcwd()
os.listdir('.')  pour son contenu
```

## Si je veux le changer :

```
os.chdir('U:')
```

## Importation de données d'un fichier .txt donc la structure est :




On voit la structure des données en 2 colonnes sans aucun entête pour présenter les données ( à supprimer éventuellement avec un éditeur de texte). La première colonne est le temps t et la seconde par exemple la tension u(t) . Le fichier data6.txt doit être dans le répertoire de travail.

```
data=np.loadtxt('data6.txt')
t=data[:,0]
u=data[:,1]
#on aurait pu taper        t,u = np.loadtxt('data6.txt')
N=t.size                   #nbre de points expérimentaux
tau=t[1]-t[0]              #intervalle de temps entre deux données successives
T=N*tau                    #durée totale de l'acquisition
fe=1.0/tau                 #fréquence d'échantillonage
```

## Travailler sur la FFT :

### FFT des données :

```
tfd=np.fft.fft(u)
spectre=np.absolute(tfd)        #le tableau tfd est à valeurs dans C
f=np.arange(N)*1.0/T            #construction des fréquences en abscisse.
```

### Graphe de la FFT :

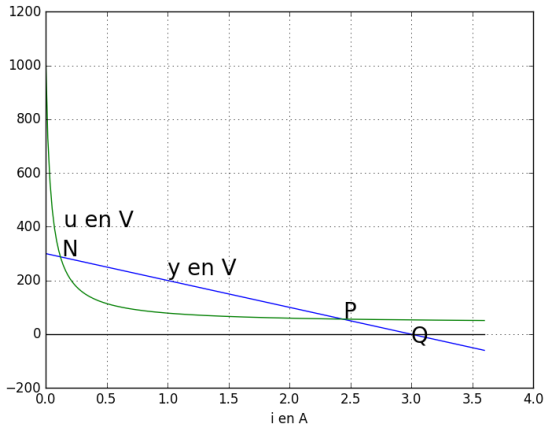
```
plt.figure()
plt.plot(f,spectre)
plt.xlabel("f en Hz")
plt.ylabel("spectre")
plt.grid()
plt.show()
```

### Sauvegarde de la FFT dans le nouveau fichier fft6.txt , la fréquence en colonne 1 et l'amplitude spectrale en colonne 2 :

```
p=open("fft6.txt",'w')          #crée le fichier en mode écriture
for i in range(0,N):
        p.write(str(f[i])), p.write("\t"), p.write(str(a[i])), p.write("\n")
                                #tabulation                    fin de ligne
p.close()
```

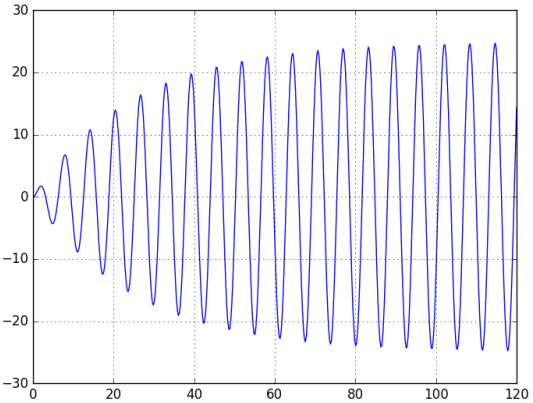
## Tracé de courbes avec textes.

```
import numpy as np
import matplotlib.pyplot as plt

i=np.linspace(0,3.6,200)
zeros=0.0*i        #axe horizontal
u=40.0+40.0/(i+0.04)
y=300.0-100.0**i
plt.plot(i,y)
plt.plot(i,zeros,'k')          #axe horizontal
plt.plot(i,u)
plt.grid()
plt.text(1,220,'y en V',fontsize=20)
plt.text(0.15,400,'u en V',fontsize=20)
plt.xlabel('i en A')
plt.text(0.13,290,'N',fontsize=20)
plt.text(2.44,60,'P',fontsize=20)
plt.text(3,-30,'Q',fontsize=20)
plt.show()
```



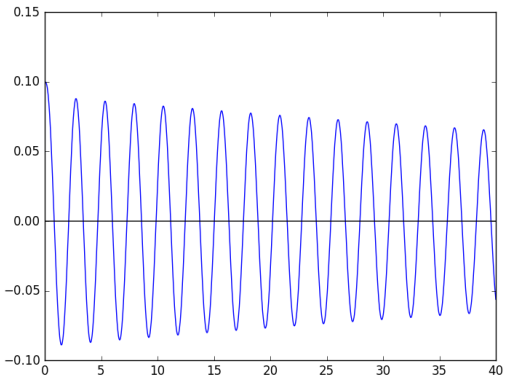

## Equa diff d'ordre 2. $\ddot{y}(t) = -y(t) - 0{,}08 \cdot \dot{y} + 2cos(t)$

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def f(y,t):
        return [y[1],-y[0]-0.08*y[1]+2*np.cos(t)]

t=np.linspace(0.0,120.0,num=500)
sol=odeint(f,[0.0,0.0],t)
plt.grid()
plt.plot(t,sol[:,0])
plt.show()
```



## Equa diff d'ordre 3. $\dddot{y}(t) = -5 \cdot \ddot{y}(t) - 6 \cdot \dot{y} - 29{,}5 \cdot y(t)$

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
def f(y,t):
        return [y[1],y[2],-5*y[2]-6*y[1]-29.5*xy[0]]

t=np.linspace(0.0,120.0,num=1200)
zeros=0.0*t        #axe horizontal
sol=odeint(f,[0.1,0.0,0.0],t)
plt.figure()
plt.plot(t,sol[:,0])
plt.plot(t,zeros,'k')          #axe horizontal
plt.show()
```



## Etude d'un système différentiel : réactions successives $A \xrightarrow{k_1} B \xrightarrow{k_2} C$.

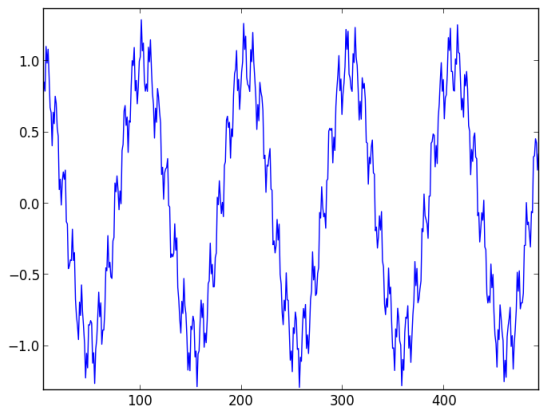
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# système différentiel
def syst_dif(Y, t, k1, k2):
   a, b, c = Y
   eq_a = -k1 * a
   eq_b = k1 * a - k2 * b
   eq_c = k2 * b
   return np.array([eq_a, eq_b, eq_c])

# conditions initiales
a_init = 1.0
b_init = 0.0
c_init = 0.0
cond_init = [a_init, b_init, c_init]

# constantes chimiques
k1 = 1.0
k2 = 1.0

# discrétisation temporelle
t_min = 0.0
t_max = 5.0
n_t = 50

tab_t = np.linspace(t_min, t_max, n_t)
tab_Y = odeint(syst_dif, cond_init, tab_t, args=(k1,k2))

# récupération des colonnes de tab_Y associées à a, b, c
a = tab_Y[:,0]
b = tab_Y[:,1]
c = tab_Y[:,2]

# tracé
plt.figure(figsize=(16,9))
plt.plot(tab_t, a, "r-", label="a")
plt.plot(tab_t, b, "g-", label="b")
plt.plot(tab_t, c, "b-", label="c")

# pour modifier la taille de textes
#plt.rc('font', size=16) #controls default text size
#plt.rc('axes', titlesize=16) #fontsize of the title
#plt.rc('axes', labelsize=16) #fontsize of the x and y labels
#plt.rc('xtick', labelsize=16) #fontsize of the x tick labels
#plt.rc('ytick', labelsize=20) #fontsize of the y tick labels
#plt.rc('legend', fontsize=24) #fontsize of the legend
plt.rcParams.update({'font.size':20})

# labels
plt.xlabel("t")
plt.ylabel("concentrations")
plt.title("Réactions successives : A -> B -> C // Évolutions temporelles des concentrations")
plt.legend()
plt.show()
```

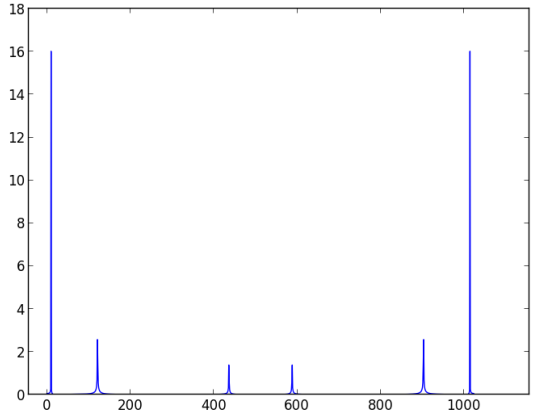
## Etude d'un système différentiel : réactions parallèles $A \xrightarrow{k_1} B$ $A \xrightarrow{k_2} C$.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# système différentiel
def syst_dif(Y, t, k1, k2):
   a, b, c = Y
   eq_a = -(k1+k2) * a
   eq_b = k1 * a
   eq_c = k2 * a
   return np.array([eq_a, eq_b, eq_c])

# conditions initiales
a_init = 1.0
b_init = 0.0
c_init = 0.0
cond_init = [a_init, b_init, c_init]

# constantes chimiques
k1 = 2.0
k2 = 1.0

# discrétisation temporelle
t_min = 0.0
t_max = 5.0
n_t = 50

tab_t = np.linspace(t_min, t_max, n_t)
tab_Y = odeint(syst_dif, cond_init, tab_t, args=(k1,k2))

# récupération des colonnes de tab_Y associées à a, b, c
a = tab_Y[:,0]
b = tab_Y[:,1]
c = tab_Y[:,2]

# tracé
plt.figure(figsize=(16,9))
plt.plot(tab_t, a, "r-", label="a")
plt.plot(tab_t, b, "g-", label="b")
plt.plot(tab_t, c, "b-", label="c")

# pour modifier la taille de textes
#plt.rc('font', size=16) #controls default text size
#plt.rc('axes', titlesize=16) #fontsize of the title
#plt.rc('axes', labelsize=16) #fontsize of the x and y labels
#plt.rc('xtick', labelsize=16) #fontsize of the x tick labels
#plt.rc('ytick', labelsize=20) #fontsize of the y tick labels
#plt.rc('legend', fontsize=24) #fontsize of the legend
plt.rcParams.update({'font.size':20})

# labels
plt.xlabel("t")
plt.ylabel("concentrations")
plt.title("Réactions parallèles : A -> B, A -> C // Évolutions temporelles des concentrations")
plt.legend()
plt.show()
```

## Traitement de Fourier manuel sans utiliser l'algorithme de la FFT.

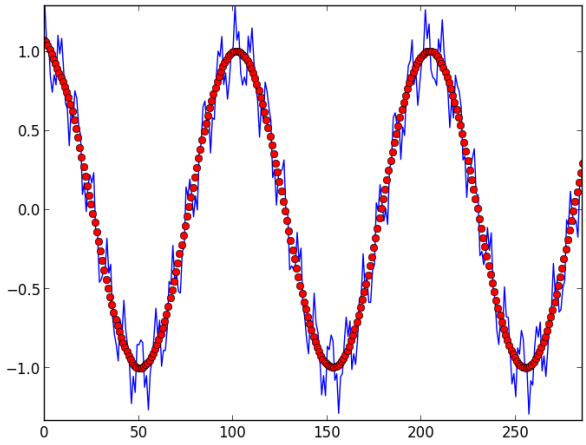
```
import numpy as np
import matplotlib.pyplot as plt

def four(ee,NN):  #définition de la transformée de Fourier numérique
        racN=np.sqrt(NN)
        ff=[0.0+0.0*1j for xx in range(0,NN)]
        for kk in range(0,NN):
                hilfe=2.0*kk*np.pi*(0.0+1j)/NN
                for pp in range(0,NN):
                        ff[kk]=ff[kk]+(np.exp(-pp*hilfe)*ee[pp])
        return np.asarray(ff/racN)

def ifour(ee,NN): #définition de la transformée de Fourier numérique inverse
        racN=np.sqrt(NN)
        ff=[0.0+0.0*1j for xx in range(0,NN)]
        for kk in range(0,NN):
                hilfe=2.0*kk*np.pi*(0.0+1j)/NN
                for pp in range(0,NN):
                        ff[kk]=ff[kk]+(np.exp(+pp*hilfe)*ee[pp])
        return np.asarray(ff/racN)


t,x,y,z = np.loadtxt("C:/signaux50ms1024.txt", unpack=True)
N=y.size
plt.figure()
plt.plot(y)
tfd=four(y,N)
plt.figure()
plt.plot(abs(tfd))
```



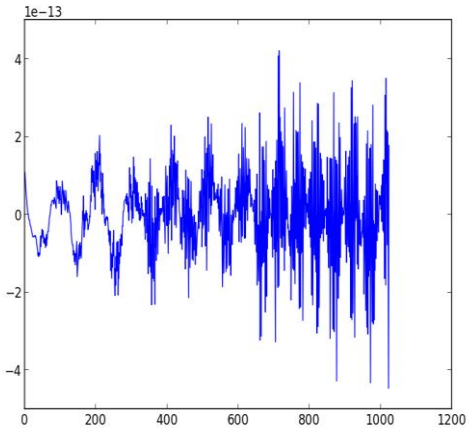




Aspect temporal : basse fréquence polluée par des HF
Aspect spectral : les deux pics de la BF sont les pics extrêmes.

```
for k in range(75,950):      #élimination des quatre pics intermédiaires
        tfd[k]=0.0

plt.figure()
plt.plot(abs(tfd))

retour=ifour(tfd,N)
plt.figure()
plt.plot(y)
plt.plot(np.real(retour),'or')
plt.figure()
plt.plot(np.imag(retour))

plt.show()
```




partie réelle de ifour(four(y)) et y




partie imaginaire de ifour(four(y))

**<u>Pour travailler avec les octets.</u>**

Utilisation des éxécutables EdHex.exe et mini_editeur.exe pour les octets d'un fichier. Mais on ne peut pas les traiter.

**Exemple de création de fichier binaire :**

```
list=[83,76,70,70]                    #liste d'entiers (en ascii  RIFF)
arr=bytearray(list)                   #création des octets associés
monfichier=open("fichier.txt ",'wb')  #ouverture de fichier en mode écriture et binaire
monfichier.write(arr)                 #écriture
monfichier.close                      #fermeture du fichier
```

**Lecture octets et réécriture :**

```
import os
import numpy as np
import matplotlib.pyplot as plt

fichier=open("la.wav",'rb')           #ouverture de fichier en mode binaire et lecture
data=fichier.read(44)                 #lecture de 44 premiers octets
N=len(data)
e=[]
for k in range(N):
   e.append(int(data[k]))             #création des entiers associés aux octets

        #traitement éventuel sur la liste d'entier. On doit recréer des entiers à la fin

octets=bytearray(e)                   #retour des entiers vers les octets
fich=open("data.txt",'wb')
fich.write(octets)
fich.close()
```