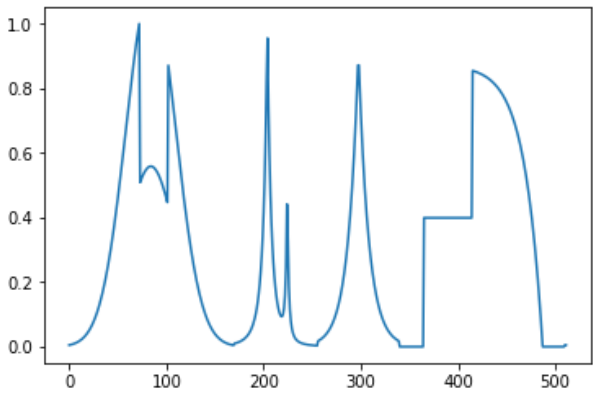
On charge un fichier .txt a l'aide de numpy

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
x= np.loadtxt('signal.txt')
N=x.size

plt.figure()
plt.plot(x)
plt.show()
```

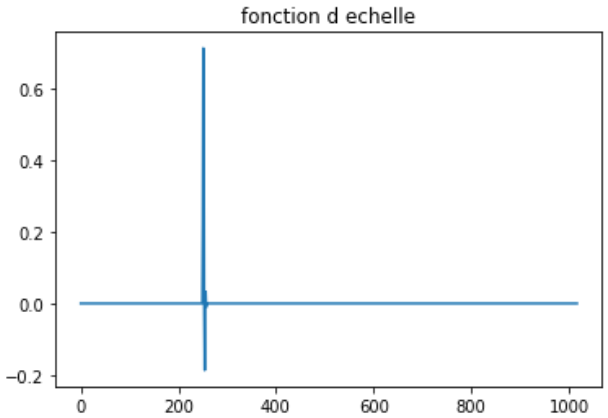


On calcule la transformée en ondelettes du signal à l'aide de la librairie pywt

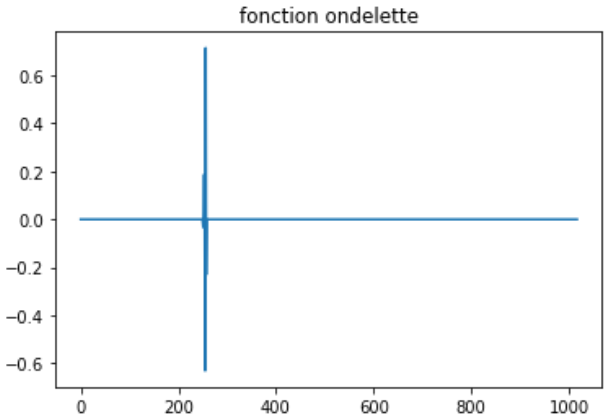
In [3]:
```
# on charge la librairie ondelettes
import pywt

ond='db4'
# on calcule l'ondelette de base
wavelet = pywt.Wavelet(ond)
```

On trace la fonction d' echelle et l'ondelette de base

In [4]:
```
Ca_e=np.zeros(N)
Cd_e=np.zeros(N)

Ca_e[128]=1

coeffs_e=[Ca_e, Cd_e]
echelle=pywt.waverec(coeffs_e,ond)

plt.figure()
plt.plot(echelle)
plt.title('fonction d echelle')
plt.show()
```



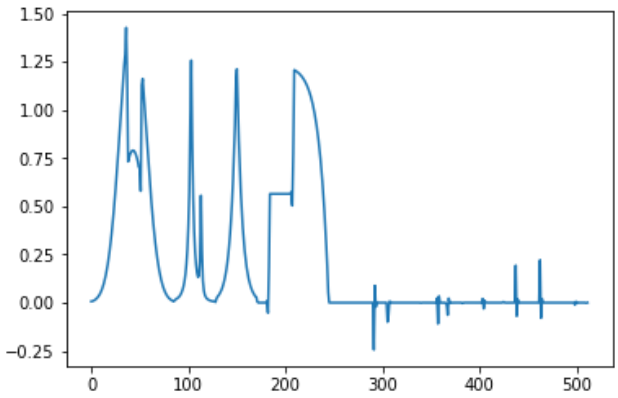
In [5]:
```
Ca_e=np.zeros(N)
Cd_e=np.zeros(N)

Cd_e[128]=1

coeffs_e=[Ca_e, Cd_e]
echelle=pywt.waverec(coeffs_e,ond)

plt.figure()
plt.plot(echelle)
plt.title('fonction ondelette')
plt.show()
```



In [6]:
```
#on calcule les coefficients de decomposition dans la base d'ondelette
Ca,Cd = pywt.dwt(x, wavelet, mode = "periodization")
```

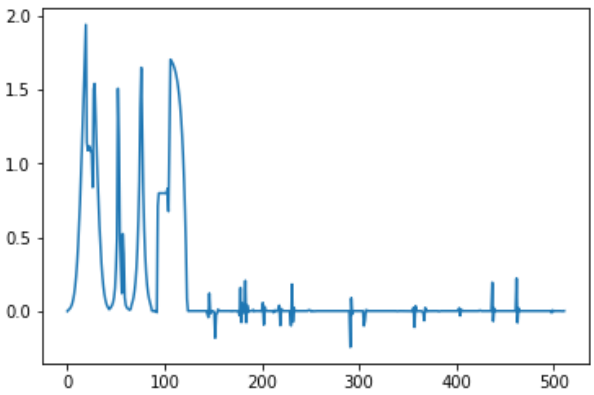
On trace les coefficients en ondelettes calculés avec l'ondelette de Daubechies

In [7]:
```
coeffs=np.concatenate((Ca,Cd))
plt.figure()
plt.plot(coeffs)
plt.show()
```



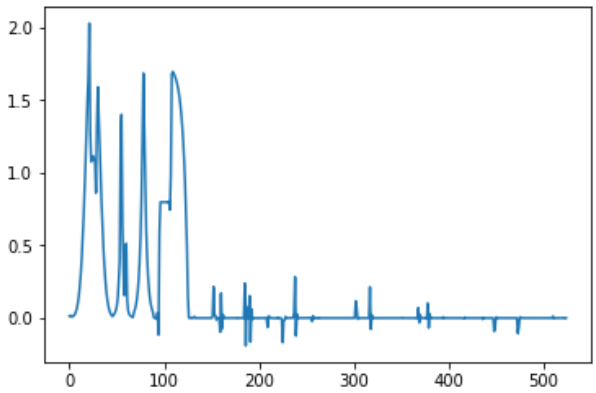
On réitère l'opération sur les coefficients d'approximation.

In [8]:
```
Caa,Cad=pywt.dwt(Ca, wavelet, mode = "periodization")

coeffs=np.concatenate((Caa,Cad,Cd))
plt.figure()
plt.plot(coeffs)
plt.show()
```



Les commandes suivantes font cette opération directement.

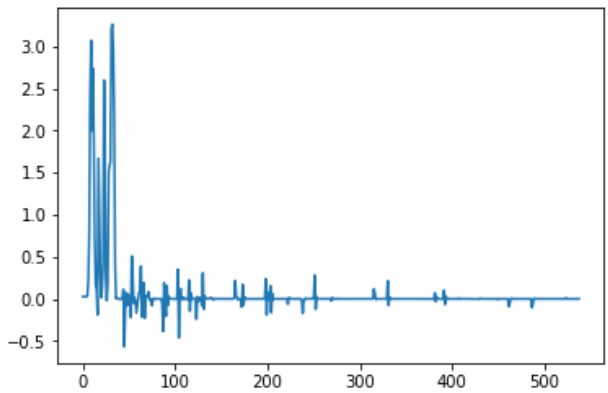
In [9]:
```
Caa,Cad,Cd = pywt.wavedec(x, ond, level=2)

coeffs=np.concatenate((Caa,Cad,Cd))
plt.figure()
plt.plot(coeffs)
plt.show()
```



On effectue une decomposition sur 4 niveaux pour ensuite chercher à obtenir une approximation parcimonieuse. L'approximation est ici de taille $512/2^4 = 2^9/2^4 = 2^5$.

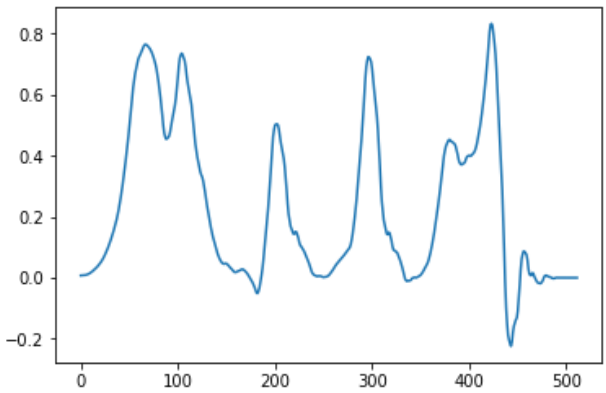
In [10]:
```
n0=4
coeffs_l=pywt.wavedec(x, ond, level=n0)
nsig=np.floor(np.log2(N))

#calcul de la taille de l'approximation
na=np.floor(nsig-n0)
na=np.int(na)
print(na)

#le resultat est une liste. On concatene pour obtenir un vecteur numpy
coeffs=np.concatenate(coeffs_l)

plt.figure()
plt.plot(coeffs)
plt.show()
```

```
5
```

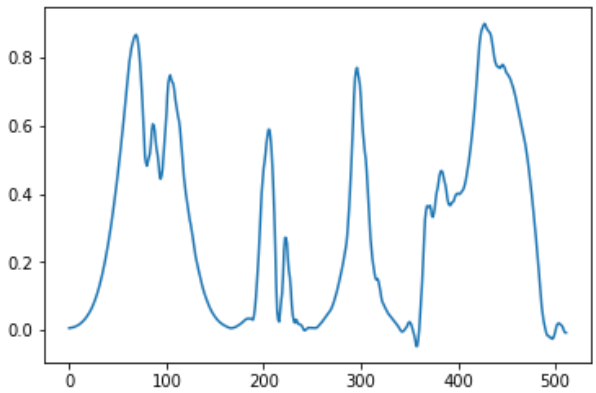


On calcule maintenant une approximation avec les M premiers vecteurs de base.

In [11]:
```
M=2**na

coeffs_M=coeffs_l.copy()
arr,sli=pywt.coeffs_to_array(coeffs_M)

#On garde seulement les M premiers coefficients et met à zéro tous les autre
s
arr_M=np.zeros(arr.size)
arr_M[0:M]=arr[0:M].copy()

#on reconstruit l'approximation du signal à partir de ses coefficients
coeffs_M=pywt.array_to_coeffs(arr_M, sli,output_format='wavedec')

x_M=pywt.waverec(coeffs_M, ond)

plt.figure()
plt.plot(x_M)
plt.show()
```

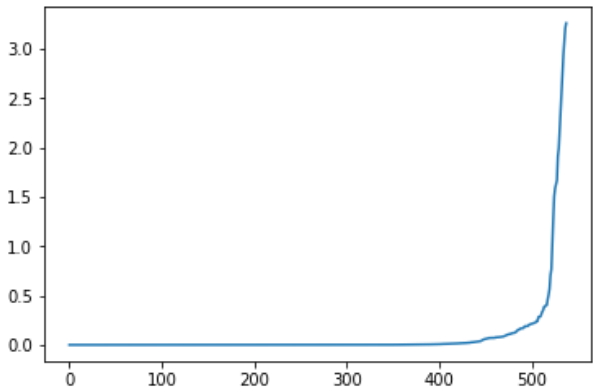


In [12]:
```
M=2**6

coeffs_M=coeffs_l.copy()
arr,sli=pywt.coeffs_to_array(coeffs_M)

#On garde seulement les M premiers coefficients et met à zéro tous les autre
s
arr_M=np.zeros(arr.size)
arr_M[0:M]=arr[0:M].copy()

#on reconstruit l'approximation du signal à partir de ses coefficients
coeffs_M=pywt.array_to_coeffs(arr_M, sli,output_format='wavedec')

x_M=pywt.waverec(coeffs_M, ond)

plt.figure()
plt.plot(x_M)
plt.show()
```



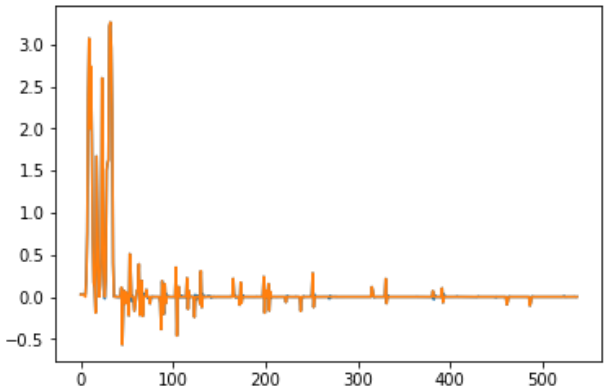
On construit maintenant une approximation non linéaire: on garde les coefficients de détail au dessus d'un certain seuil.

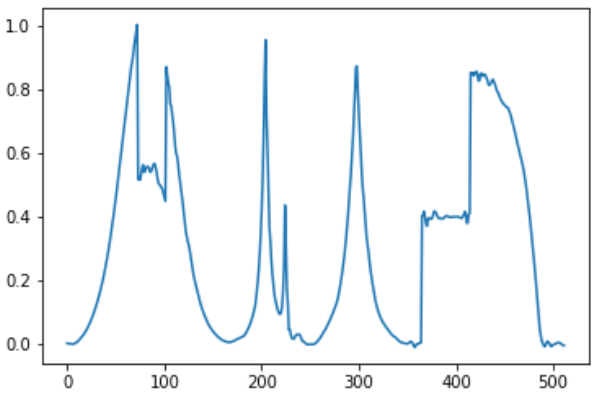
In [13]:
```
arr_T=arr.copy()

#On met en mémoire l'approximation qu'on ne touchera pas.
arr_Ta=arr[0:na].copy()

#On met a zero l'approximation
arr_T[0:na]=np.zeros(na)

arr_Ts=np.sort(np.abs(arr_T))
arg_Ts=np.argsort(np.abs(arr_T))

plt.figure()
plt.plot(arr_Ts)
plt.show()
```



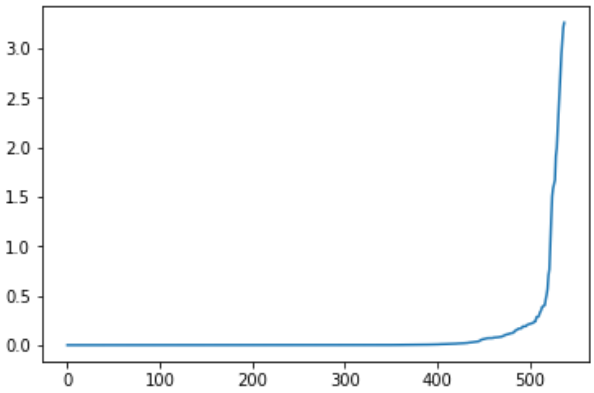
On garde les M coefficients les plus gros

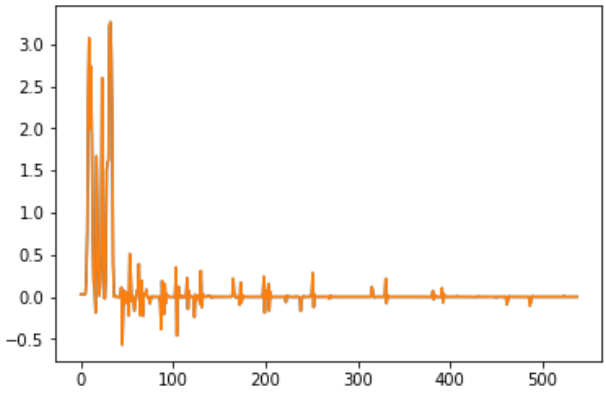
In [14]:
```
M=2**6

arr_T[arg_Ts[0:N-M]]=np.zeros(N-M)

arr_T[0:na]=arr_Ta

plt.figure()
plt.plot(arr)
plt.plot(arr_T)
plt.show()

#on reconstruit l'approximation du signal à partir de ses coefficients
coeffs_nT=pywt.array_to_coeffs(arr_T, sli)

x_nT=pywt.waverecn(coeffs_nT, ond)

plt.figure()
plt.plot(x_nT)
plt.show()
```

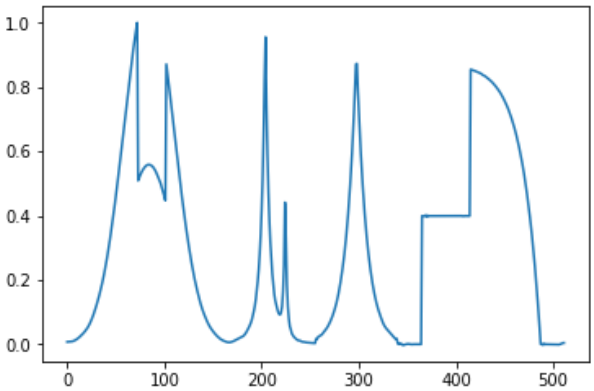




In [15]:
```
M=2**7

arr_T=arr.copy()

#On met en mémoire l'approximation qu'on ne touchera pas.
arr_Ta=arr[0:na].copy()

#On met a zero l'approximation
arr_T[0:na]=np.zeros(na)

arr_Ts=np.sort(np.abs(arr_T))
arg_Ts=np.argsort(np.abs(arr_T))

plt.figure()
plt.plot(arr_Ts)
plt.show()

arr_T[arg_Ts[0:N-M]]=np.zeros(N-M)

arr_T[0:na]=arr_Ta

plt.figure()
plt.plot(arr)
plt.plot(arr_T)
plt.show()

#on reconstruit l'approximation du signal à partir de ses coefficients
coeffs_nT=pywt.array_to_coeffs(arr_T, sli,output_format='wavedec')

x_nT=pywt.waverec(coeffs_nT, ond)

plt.figure()
plt.plot(x_nT)
plt.show()
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
100
200
300
400
500


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500