On importe les bibliotheques dont on va se servir

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

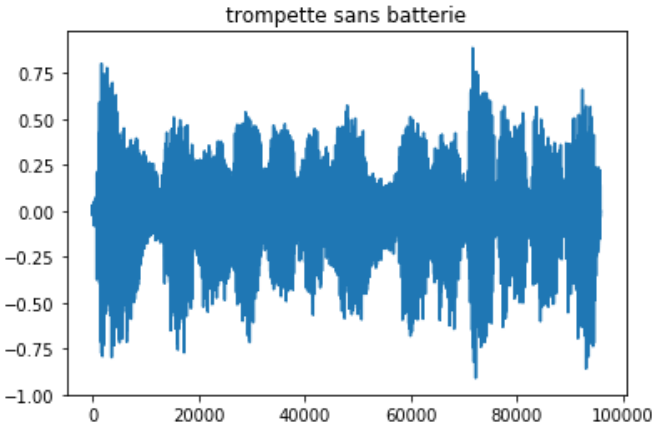
1. Cas des sons

On importe le signal

In [2]:
```
fs, tromp = wavfile.read('trompEssen.wav')
N=tromp.size

#on normalise le signal
m=np.max(np.abs(tromp))
tromp=tromp/(1.1*m)
```

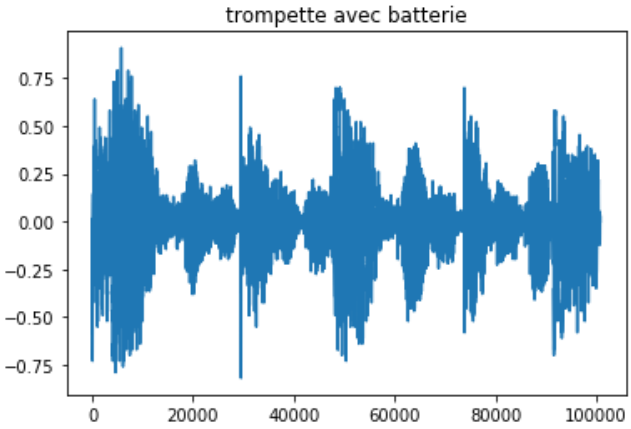
On trace le signal.

In [3]:
```
plt.figure(1)
plt.plot(tromp)
plt.title('trompette sans batterie')
plt.show()
```



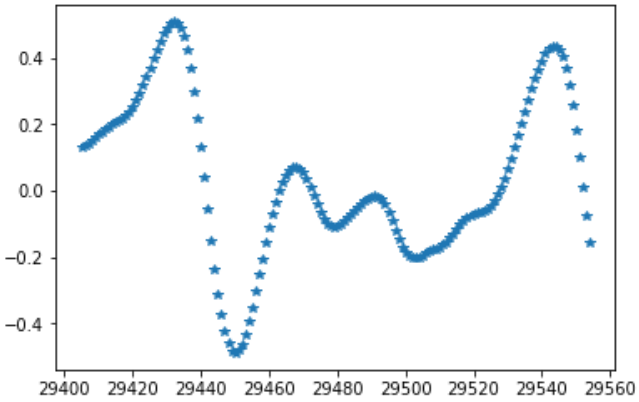
Un autre extrait du morceau

In [4]:
```
fs, trompb = wavfile.read('trompbattEssen.wav')
Nb=trompb.size

m=np.max(np.abs(trompb))
trompb=trompb/(1.1*m)

plt.figure(2)
plt.plot(trompb)
plt.title('trompette avec batterie')
plt.show()
```



On zoome sur une partie du signal

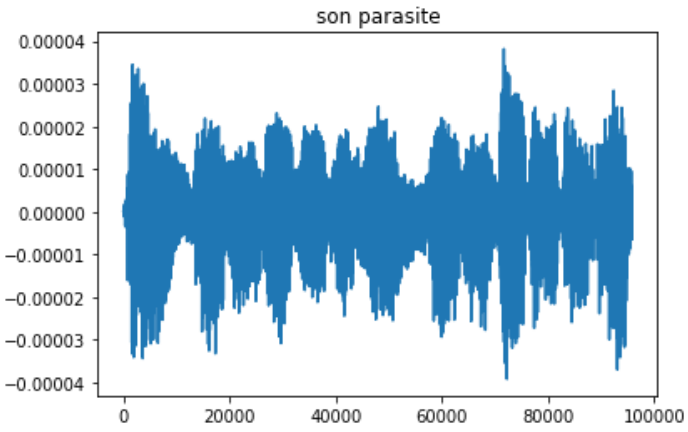
In [5]:
```
n0=29505
plt.figure(3)
plt.plot(np.arange(n0-100,n0+50,1),tromp[n0-100:n0+50],'*')
plt.show()
```



On veut régler des problèmes de traitement du signal: comment enlever le son parasite ?

In [6]:
```
fs1, son1 = wavfile.read('son1.wav')
N=son1.size

#on normalise le signal
m=np.max(np.abs(son1))
son1=tromp/(1.1*m)
```

In [7]:
```
plt.figure(4)
plt.plot(son1)
plt.title('son parasite')
plt.show()
```



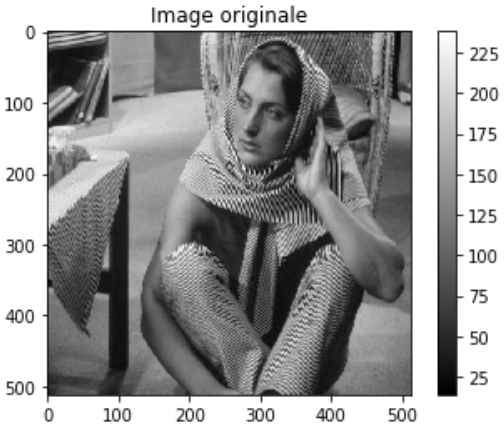
2. Cas des images

Une image noire et blanc est un tableau de nombres: chaque pixel donne une intensité de gris qui est codé par un nombre.

In [8]:
```
from scipy import ndimage as ndim
#on charge l'image de lena à l'aide de la fonction imread de la bibliothèque nd
image de scipy
y =ndim.imread("barb.bmp")
#l'image est de type int on la cast en float pour pouvoir faire des calculs
y=y.astype(float)
[N1,N2]=y.shape
```

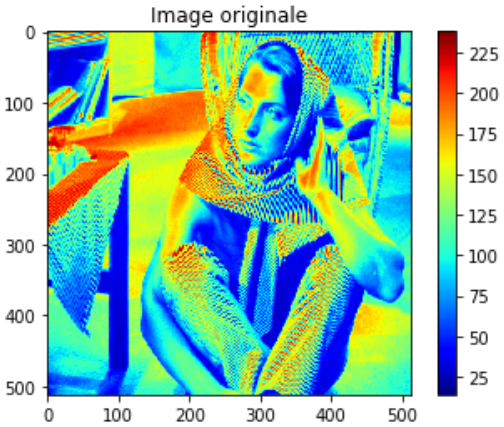
On affiche l'image

In [9]:
```
import matplotlib.pyplot as plt
#on affiche l'image à l'aide de la fonction imshow de la bibliothèque pyplot de matplotlib
#on donne un numero à la figure
plt.figure(7)
#on l'affiche a l'aide du code RGB
plt.imshow(y,cmap="gray")
#on affiche la colorbar associée
plt.colorbar()
#on lui donne un titre
plt.title('Image originale')
#on affiche tout cela sur la figure 1
plt.show()
```



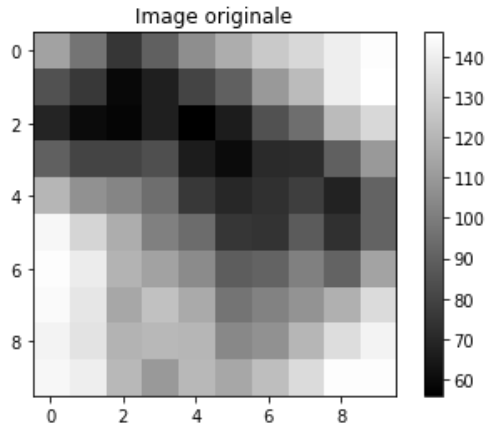
On change de code couleur pour la visualiser.

In [10]:
```
import matplotlib.pyplot as plt
#on affiche l'image à l'aide de la fonction imshow de la bibliothèque pyplot de matplotlib
#on donne un numero à la figure
plt.figure(8)
#on l'affiche a l'aide du code RGB
plt.imshow(y,cmap="jet")
#on affiche la colorbar associée
plt.colorbar()
#on lui donne un titre
plt.title('Image originale')
#on affiche tout cela sur la figure 1
plt.show()
```



on zoome sur l'oeil gauche et on regarde les valeurs associées

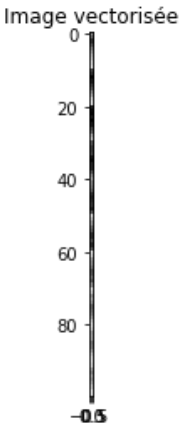
In [11]:
```
n1=100
n2=275
selx = np.arange(n1-5,n1+5)
sely = np.arange(n2-5,n2+5)

plt.figure(10)
plt.imshow(y[n1-5:n1+5,n2-5:n2+5],cmap="gray")
#on affiche la colorbar associée
plt.colorbar()
#on lui donne un titre
plt.title('Image originale')
#on affiche tout cela sur la figure 1
plt.show()

print(y[n1-5:n1+5,n2-5:n2+5])
```




```
[[ 113.   97.   75.   90.  106.  117.  127.  132.  140.  145.]
 [  85.   76.   59.   67.   80.   90.  110.  122.  140.  146.]
 [  69.   60.   58.   67.   56.   66.   85.   95.  122.  132.]
 [  90.   80.   80.   84.   66.   60.   71.   72.   90.  110.]
 [ 120.  107.  103.   95.   76.   70.   73.   78.   68.   91.]
 [ 143.  131.  117.  101.   94.   75.   74.   88.   73.   91.]
 [ 145.  139.  119.  113.  105.   89.   91.  101.   91.  114.]
 [ 144.  137.  115.  124.  116.   97.  102.  108.  118.  133.]
 [ 142.  136.  119.  121.  120.  104.  107.  120.  134.  142.]
 [ 143.  140.  121.  110.  121.  115.  123.  133.  145.  145.]]
```

On peut transformer facilement le tableau en un vecteur.

In [12]:
```
y1=y[n1-5:n1+5,n2-5:n2+5].copy()
u=y1.reshape(100,1)

plt.figure(10)
plt.imshow(u,cmap="gray")
#on affiche la colorbar associée
#on lui donne un titre
plt.title('Image vectorisée')
#on affiche tout cela sur la figure 1
plt.show()

print(u[0:20])
```



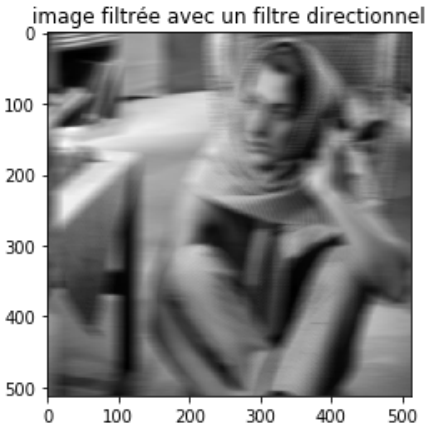

```
[[ 113.]
 [  97.]
 [  75.]
 [  90.]
 [ 106.]
 [ 117.]
 [ 127.]
 [ 132.]
 [ 140.]
 [ 145.]
 [  85.]
 [  76.]
 [  59.]
 [  67.]
 [  80.]
 [  90.]
 [ 110.]
 [ 122.]
 [ 140.]
 [ 146.]]
```

Ici on a une image dégradée.

In [13]:
```
def uniformx(n,s):
    h=np.zeros((n,n))
    h[0,0:int(np.ceil(s/2))] = 1
    h = h/np.sum(h[:])
    return h
```

In [14]:
```
hu=uniformx(N1,50);
```

In [15]:
```
#calcul de la fonction de transfert du filtre gaussien
huchap=np.fft.fft2(hu);

#filtrage de l'image
ychap=np.fft.fft2(y)

#filtrage par le filtre gaussien
yfiltuchap=ychap*huchap

yfiltu=np.real(np.fft.ifft2(yfiltuchap))

plt.figure(4)
plt.imshow(yfiltu,cmap='gray')
plt.title('image filtrée avec un filtre directionnel')
plt.show()
```



Objectif d'un ingénieur en traitement du signal: restaurer l'image ! Et l'objectif du cours est d'étudier les outils mathématiques qui vont permettre de travailler à cette restauration.