Importation de l'image à partir de la bibliothèque ndimage de scipy

In [1]:
```
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 double pour pouvoir faire des calculs
y=y.astype(float)
#la taille de l'image peut s'obtenir à l'aide de l'attribut shape
[n1,n2]=y.shape
```

Visualisation de l image

In [2]:
```
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(1)
#on l'affiche en niveau de gris
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()
```

In [3]:
```
#pour cela nous avons besoin de la bibliothèque numpy
import numpy as np
```

Définition de la réponse impulsionnelle du filtre

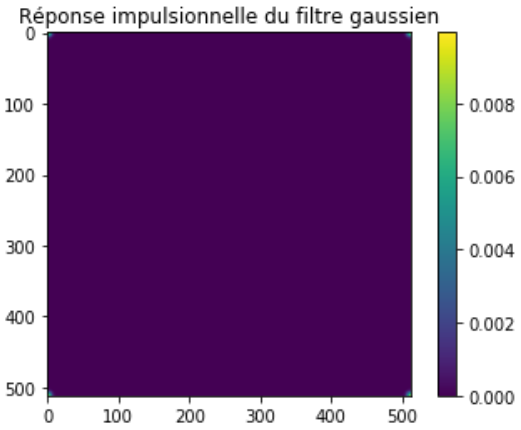
In [4]:
```
def gaussian(n,s):
    x = np.concatenate((np.arange(0,n/2,1),np.arange(-n/2,0,1)))
    [Y,X] = np.meshgrid(x,x)
    h = np.exp((-X**2-Y**2)/(2*s**2))
    h = h/np.sum(h[:])
    return h
```

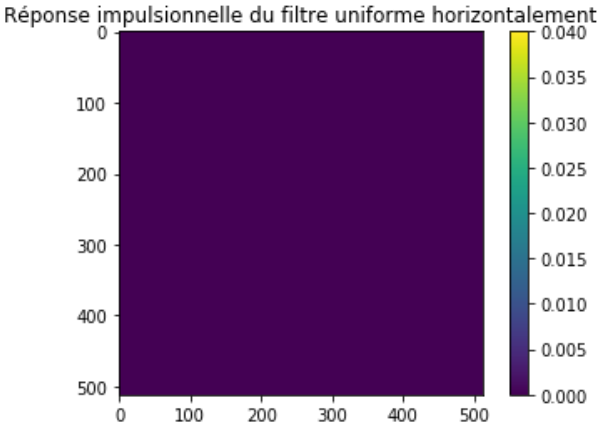
In [5]:
```
def uniform(n,s):
    h=np.zeros((n,n))
    h[0:int(np.ceil(s/2)),0:int(np.ceil(s/2))] = 1
    h[n-int(np.floor(s/2)):n,0:int(np.ceil(s/2))]=1
    h[0:int(np.ceil(s/2)),n-int(np.floor(s/2)):n]=1
    h[n-int(np.floor(s/2)):n,n-int(np.floor(s/2)):n]=1
    h = h/np.sum(h[:])
    return h
```

In [6]:
```
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
```

On visualise les filtres

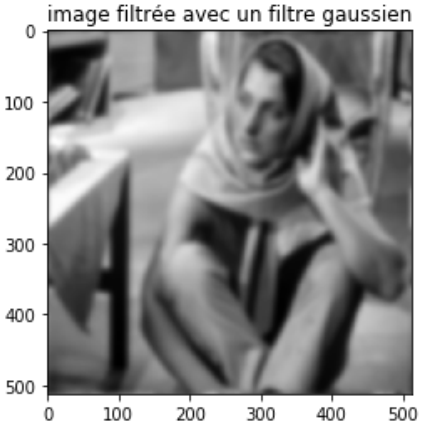
In [7]:
```
hg=gaussian(n1,4);
plt.figure(2)
plt.imshow(hg)
plt.colorbar()
plt.title('Réponse impulsionnelle du filtre gaussien')
plt.show()

hu=uniformx(n1,50);
plt.figure(3)
plt.imshow(hu)
plt.colorbar()
plt.title('Réponse impulsionnelle du filtre uniforme horizontalement')
plt.show()
```





On passe en Fourier pour effectuer le filtrage.

In [8]:
```
#calcul de la fonction de transfert du filtre gaussien
hgchap=np.fft.fft2(hg);

#calcul de la fonction de transfert du filtre uniforme
huchap=np.fft.fft2(hu);

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

#filtrage par le filtre gaussien
yfiltgchap=ychap*hgchap

yfiltg=np.real(np.fft.ifft2(yfiltgchap))

plt.figure(4)
plt.imshow(yfiltg,cmap='gray')
plt.title('image filtrée avec un filtre gaussien')
plt.show()
#filtrage par le filtre uniforme horizontal
yfiltuchap=ychap*huchap

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

plt.figure(5)
plt.imshow(yfiltu,cmap='gray')
plt.title('avec un filtre uniforme horizontalement')
plt.show()
```



In [9]:
```
def diffy(n):
    h=np.zeros((n,n))
    h[0,0] = 1
    h[0,1]=-1
    h = h/np.sum((np.abs(h))[:])
    return h
```

In [10]:
```
hl=diffy(n1);
plt.figure(6)
plt.imshow(hu)
plt.colorbar()
plt.title('Réponse impulsionnelle du filtre differentiateur')
plt.show()

hlchap=np.fft.fft2(hl);

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

#filtrage par le filtre differentiateur
yfiltlchap=ychap*hlchap
yfiltl=np.real(np.fft.ifft2(yfiltlchap))

plt.figure(7)
plt.imshow(yfiltl,cmap='gray')
plt.title('avec un filtre differentiateur')
plt.show()
```