Compte Rendu de TP 03

Analyse et Traitement des images

Nom: BOUARICHE

Prénom: Iheb

Année : 2023/2024

Spécialité : Instrumentation an2

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d
import cv2
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from skimage import io
```

Exercice 01:

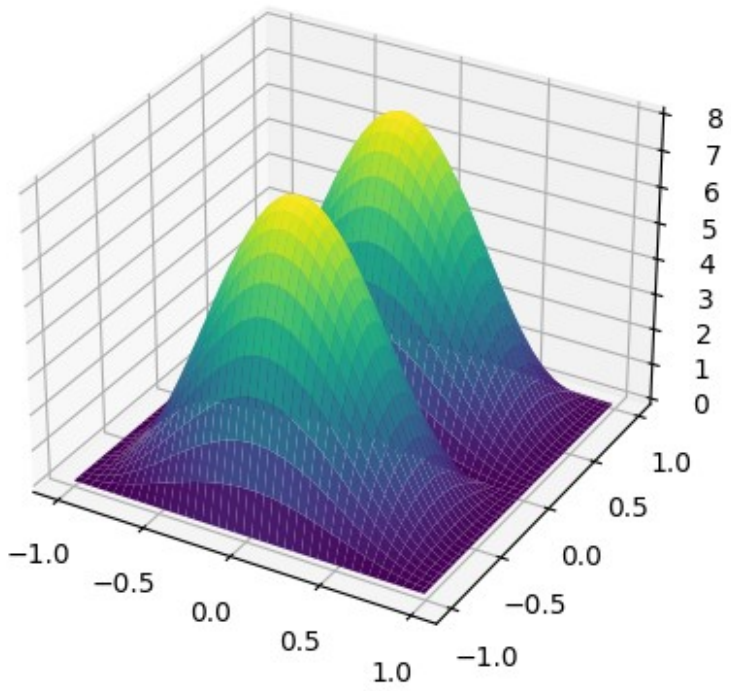
Question 1:

J'ai pas pu utiliser l'instruction signal.freqz, c'est parceque elle est utilisable pour les filtre de 1d et non pas des images de 2d sur GOOGLE COLAB.

```
# Je vais definir une fonction freqz qui va calculer la réponse fréquentielle d'un signal 2d (image)
def freqz(mask):
  img = np.zeros((256,256))
  img[127:130,127:130]=mask
  a = [i for i in range(-128,128)]
  x= []
  y= []
  y,x = np.meshgrid(a,a) # x et y sont entre -128 a 128
  x = x/128
  y = y/128
  z = np.abs(np.fft.fftshift(np.fft.fft2(img))) # l'application de la Transformé de fourrier puis le shift puis le abs.
  return x,y,z
```

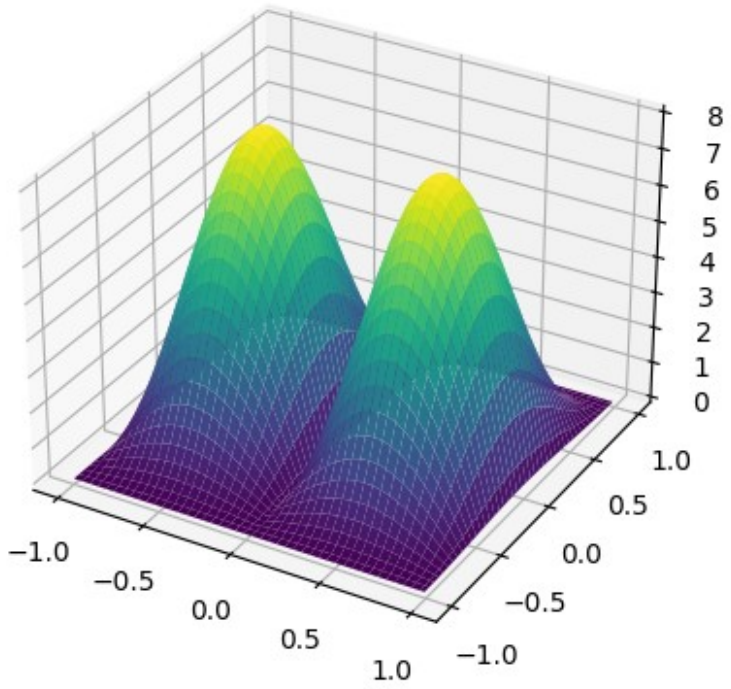
Le filtre de Sobel

```
S1 = np.array([[1,0,-1],[2,0,-2],[1,0,-1]])
S2 = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])

x,y,Z1 = freqz(S1)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis", edgecolor = "none")
plt.show()
```



Le Noyeau S1 est un filtre passe haut selon X et un filtre passe bas selon Y.

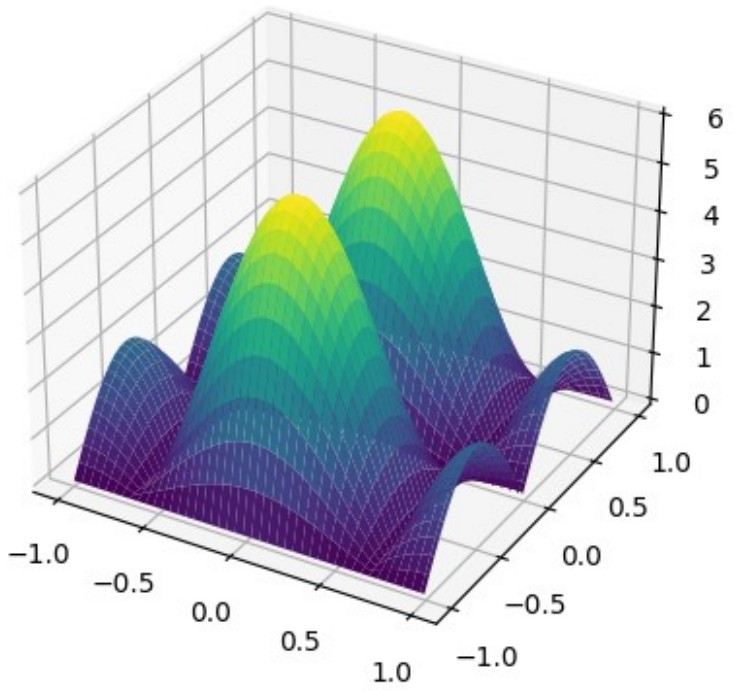
```
x,y,Z2 = freqz(S2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z2, cmap = "viridis", edgecolor = "none")
plt.show()
```



Le noyeau S2 est un filtre passe haut selon y et un filtre passe-bas selon X.

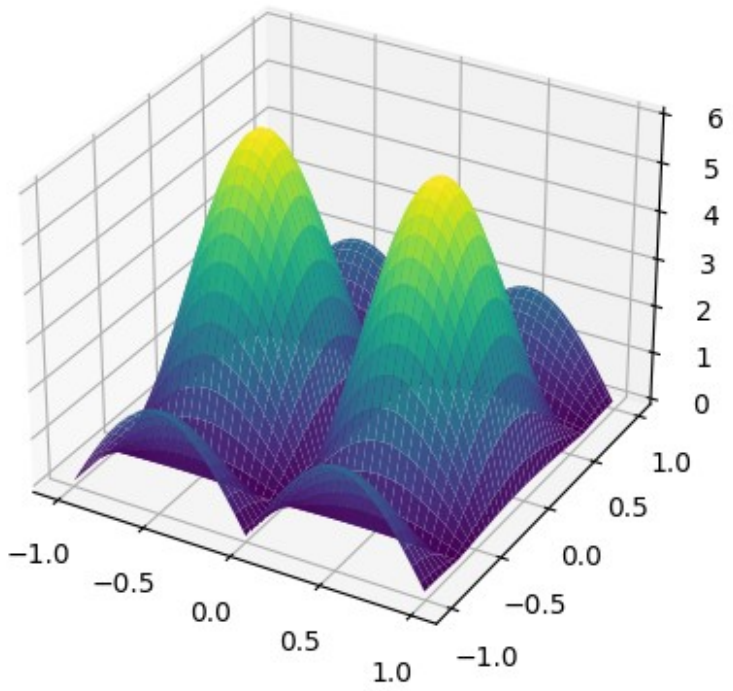
Filtre de Prewitt

```
S1 = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
S2 = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])

x,y,Z1 = freqz(S1)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis", edgecolor = "none")
plt.show()
```



On a un filtre passe haut selon x et un filtre passe bas selon y

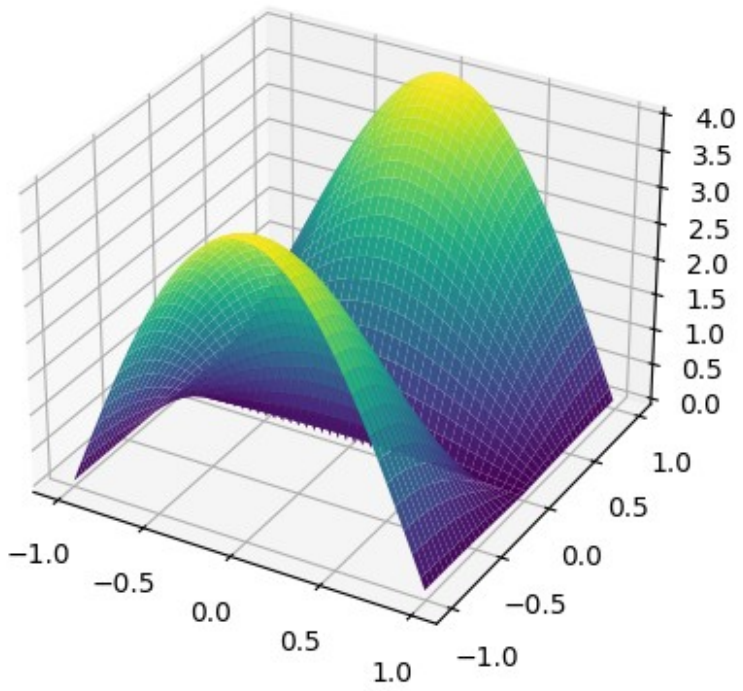
```
x,y,Z2 = freqz(S2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z2, cmap = "viridis", edgecolor = "none")
plt.show()
```



On a un filtre passe haut selon y et un filtre passe bas selon x

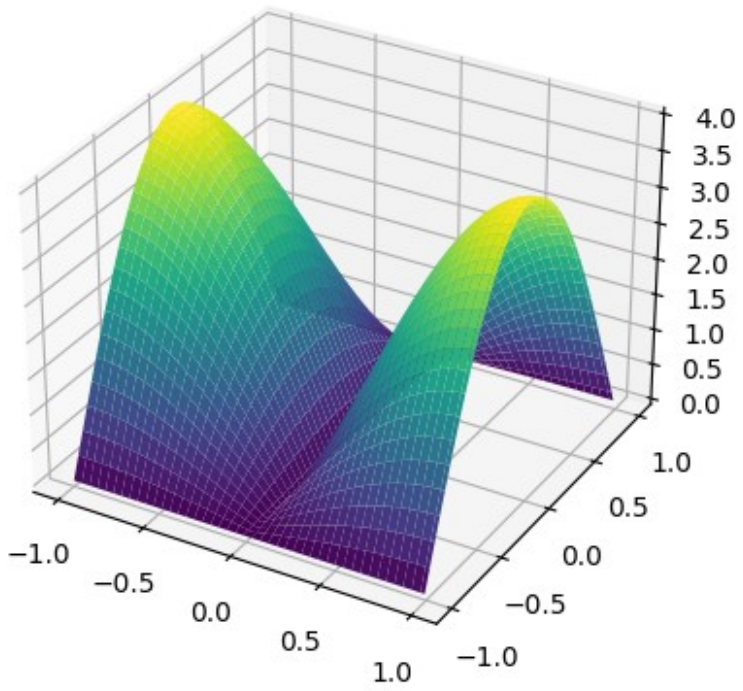
Filtre de Robert

```
def freqz2(mask):
  img = np.zeros((256,256))
  img[128:130,128:130]=mask
  a = [i for i in range(-128,128)]
  y,x = np.meshgrid(a,a) # x et y sont entre -128 a 128
  x = x/128
  y = y/128
  spm = np.abs(np.fft.fftshift(np.fft.fft2(img)))
  return x,y,spm

S1 = np.array([[1,-1],[1,-1]])
S2 = np.array([[1,1],[-1,-1]])

x,y,Z1 = freqz2(S1)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis", edgecolor = "none")
plt.show()
```



Filtre passe-haut selon x et filtre passe-bas selon y

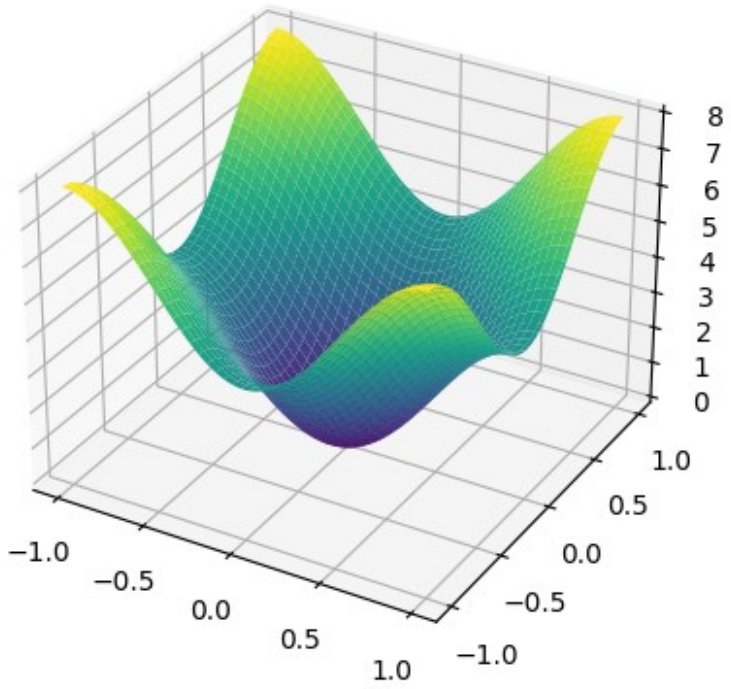
```
x,y,Z2 = freqz2(S2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z2, cmap = "viridis", edgecolor = "none")
plt.show()
```



Filtre passe-haut selon x et filtre passe-bas selon y

Filtre de Laplace

```
S1 = np.array([[0,1,0],[1,-4,1],[0,1,0]])

x,y,Z1 = freqz(S1)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis", edgecolor = "none")
plt.show()
```



Ce filtre est invariant pour les rotations et sensible au bruit.

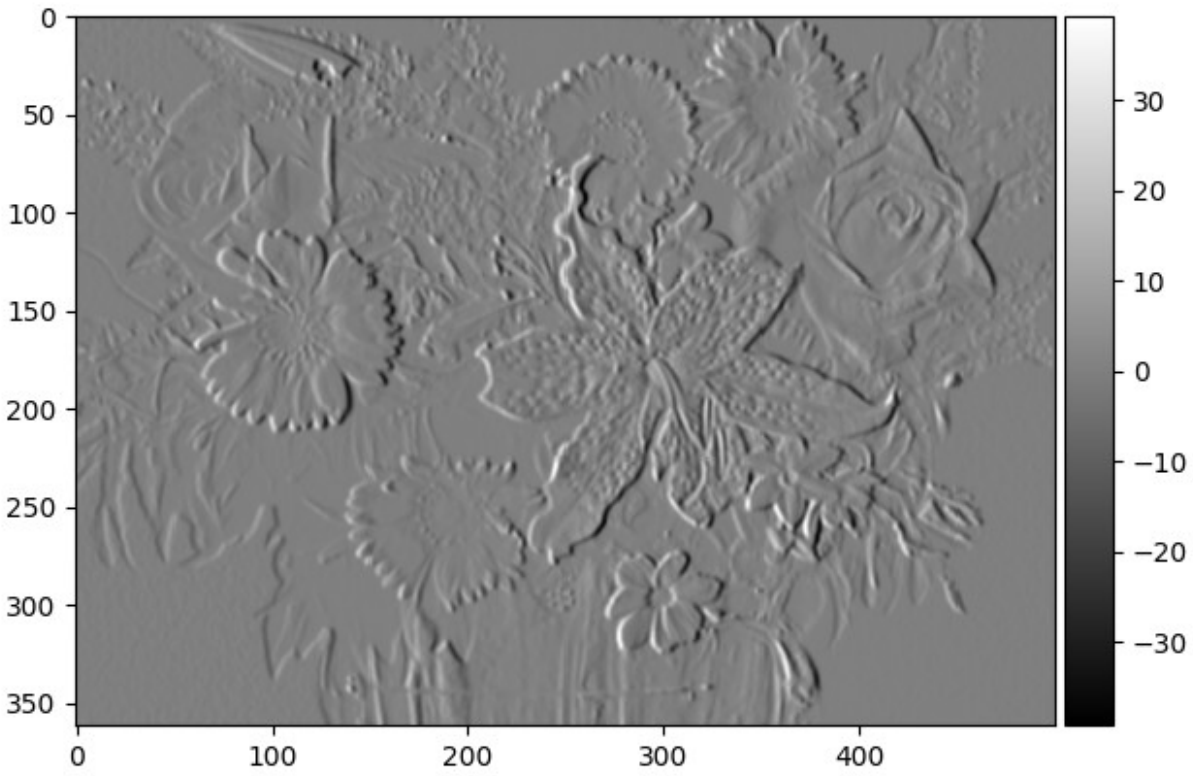
Question 2:

```
img = io.imread("flowers.tif",True)
io.imshow(img,cmap="gray")
<matplotlib.image.AxesImage at 0x7cfc73b9cb50>
```

Filtre de Sobel

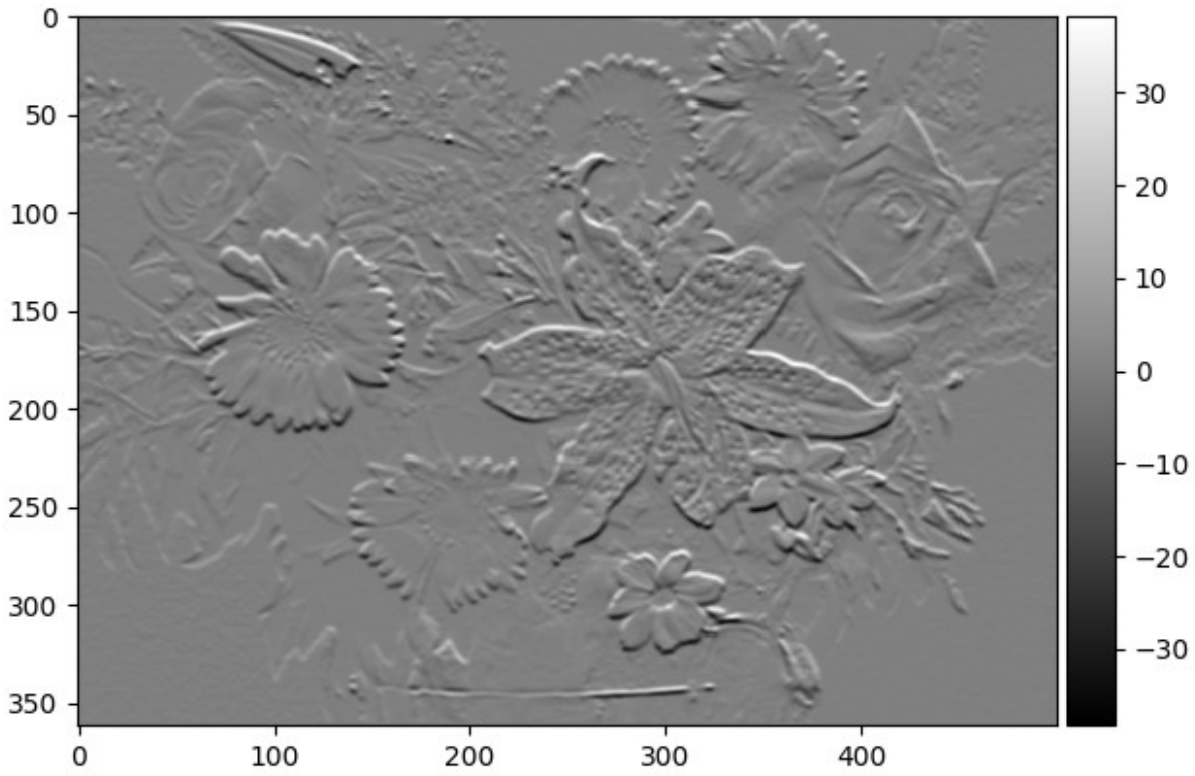
```
sobelx = cv2.Sobel(src=img,ddepth=cv2.CV_64F,dx=1,dy=0,ksize=5)
sobely = cv2.Sobel(src=img,ddepth=cv2.CV_64F,dx=0,dy=1,ksize=5)

io.imshow(sobelx,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc77996d10>
```




```
io.imshow(sobely,cmap="gray")
```

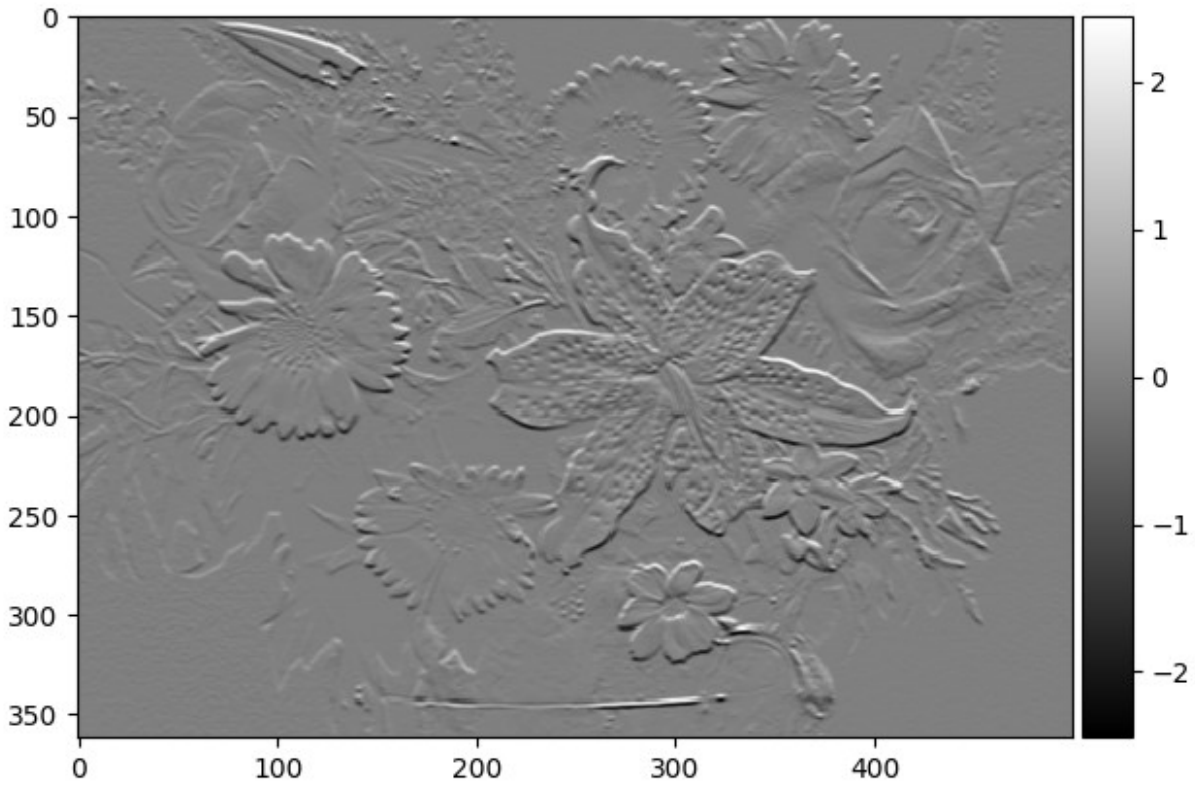
```
<matplotlib.image.AxesImage at 0x7cfc73f6d510>
```



Filtre de Prewitt

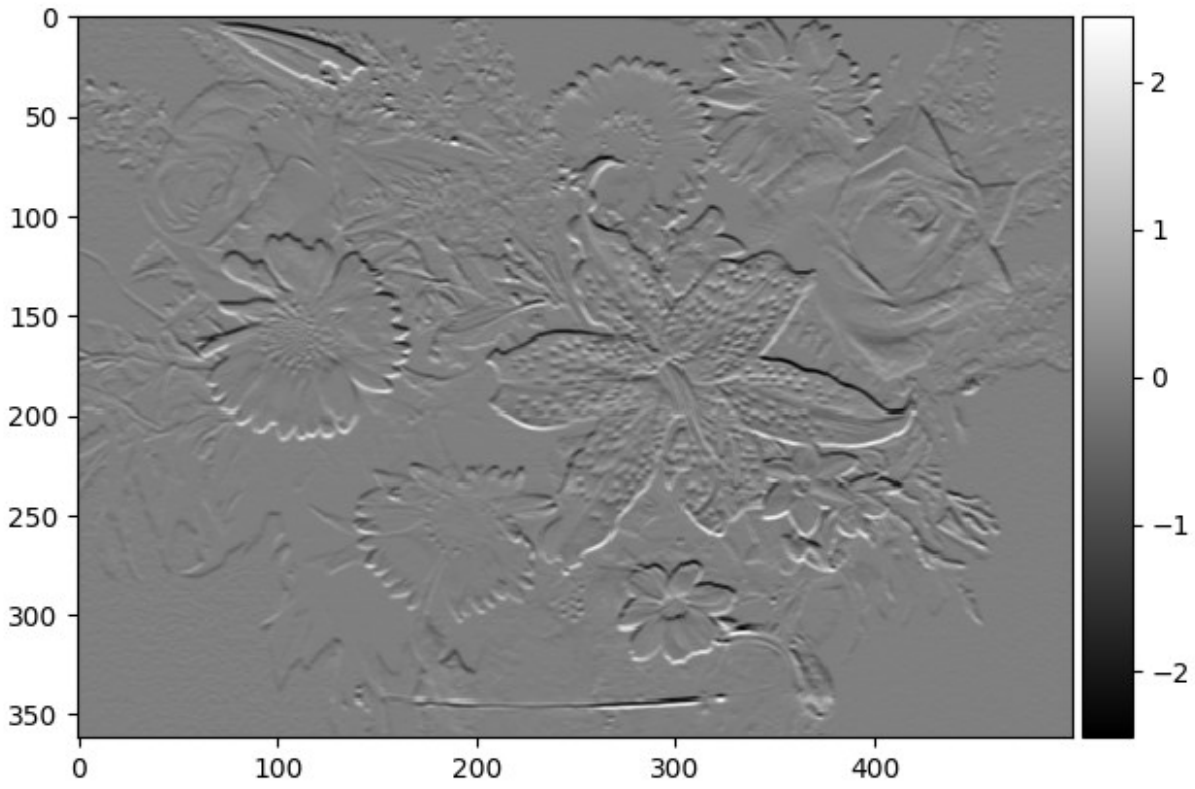
```
kernel_prewitt_x = np.array([[-1, -1, -1],[0, 0, 0],[1, 1, 1]])
kernel_prewitt_y = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
Prewittx = cv2.filter2D(img, cv2.CV_64F, kernel_prewitt_x)
Prewitty = cv2.filter2D(img, cv2.CV_64F, kernel_prewitt_y)

io.imshow(Prewittx,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7429a170>
```




```
io.imshow(Prewitty,cmap="gray")
```

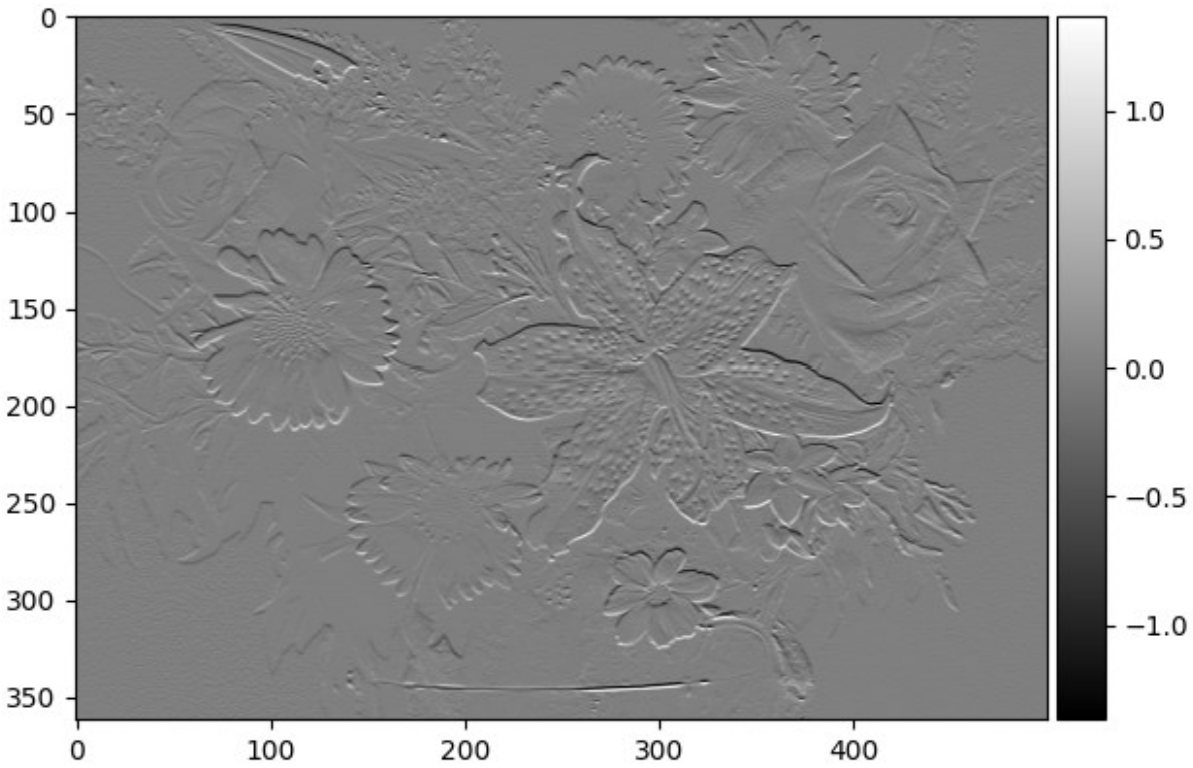
```
<matplotlib.image.AxesImage at 0x7cfc74486350>
```



Filtre de Robert

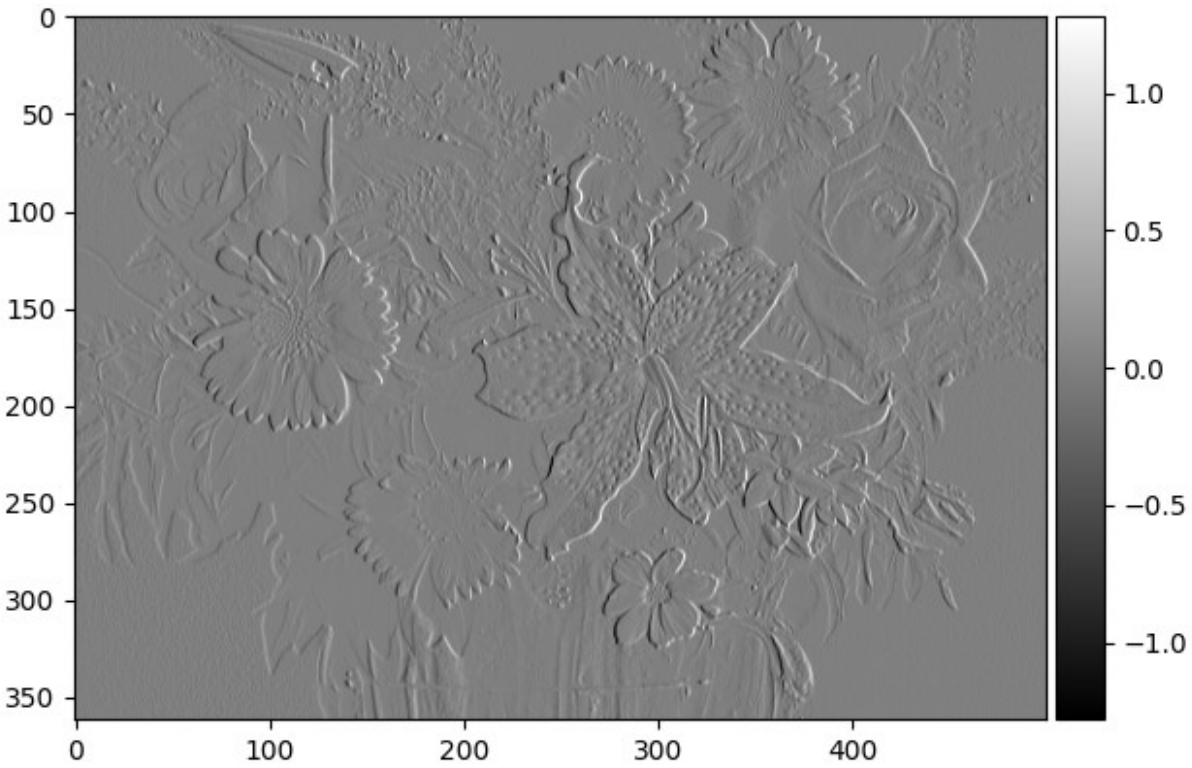
```
S1 = np.array([[1,1],[-1,-1]])
S2 = np.array([[1,-1],[1,-1]])
Robertx = cv2.filter2D(img, cv2.CV_64F, S1)
Roberty = cv2.filter2D(img, cv2.CV_64F, S2)

io.imshow(Robertx,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7488eda0>
```




```
io.imshow(Roberty,cmap="gray")
```

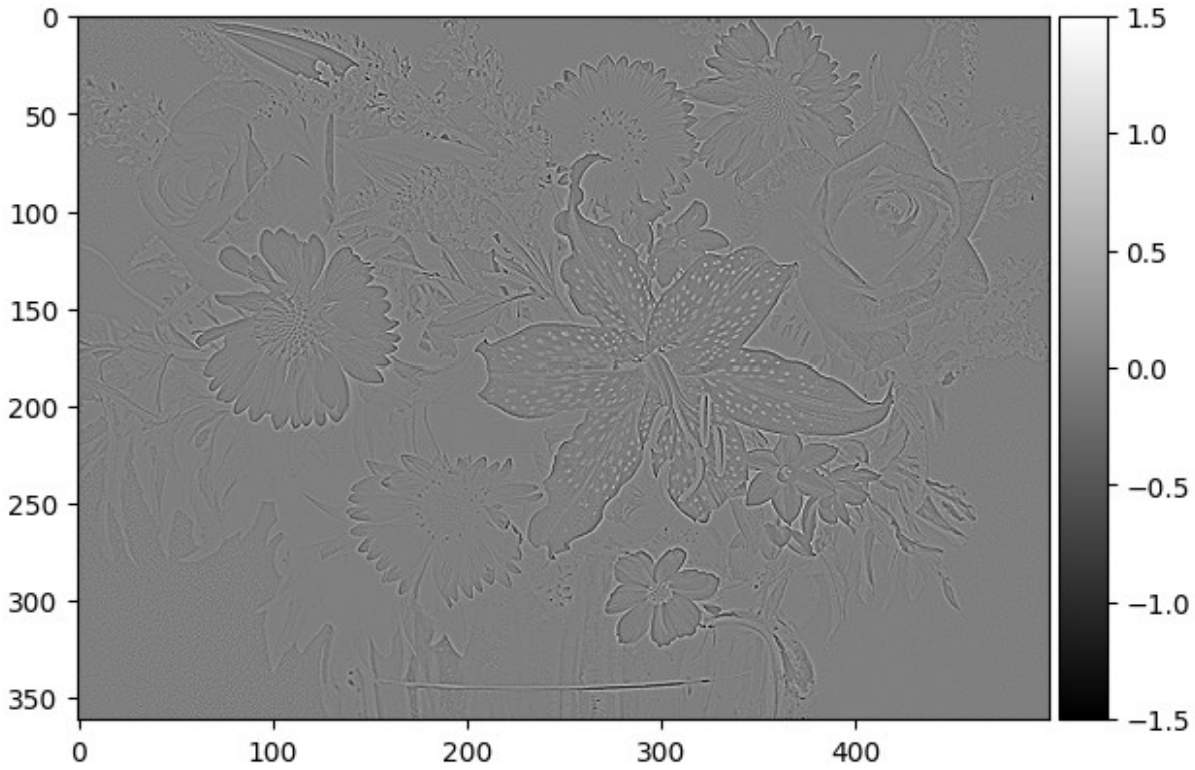
```
<matplotlib.image.AxesImage at 0x7cfc79c9b850>
```



Filtre de Laplace

```
S1 = np.array([[0,1,0],[1,-4,1],[0,1,0]])
Laplace = cv2.filter2D(img, cv2.CV_64F, S1)

io.imshow(Laplace,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc77191720>
```



Question 03: Le seuil peut avoir un impact significatif sur la qualité de la détection des contours, car un seuil trop bas peut conduire à la détection de bruit, tandis qu'un seuil trop élevé peut conduire à la perte de contours importants. Il est souvent nécessaire de régler expérimentalement le seuil pour obtenir les meilleurs résultats en fonction de la nature spécifique de l'image.

Exercice 02

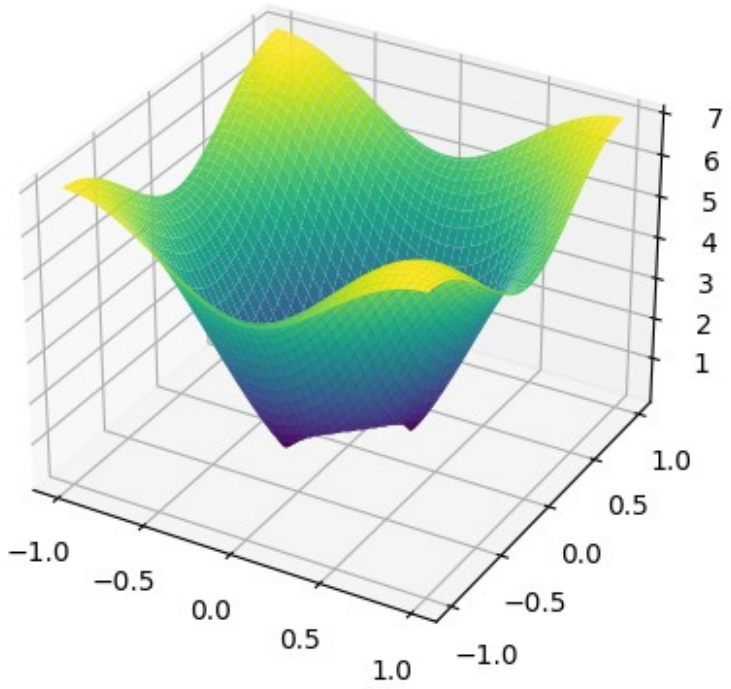
```
S = ([[0,1,1],[1,-4,1],[0,1,0]])
print("La réponse impulsionnelle h(x,y) =",S)
```

```
La réponse impulsionnelle h(x,y) = [[0, 1, 1], [1, -4, 1], [0, 1, 0]]
```

```
x,y,Z1 = freqz(S)
print("La réponse fréquentielle du filtre h(x,y) =",Z1)
```

```
La réponse fréquentielle du filtre h(x,y) = [[7.         6.99974184
6.99896746 ... 6.99767715 6.99896746 6.99974184]
 [6.99974184 6.99879527 6.99733277 ... 6.99948015 7.00008465
7.00017179]
 [6.99896746 6.99733277 6.99518182 ... 7.00076257 7.0006833
7.00008465]
 ...
 [6.99767715 6.99948015 7.00076257 ... 6.98916183 6.99251607
6.99535518]
```

```
 [6.99896746 7.00008465 7.0006833  ... 6.99251607 6.99518182
6.99733277]
 [6.99974184 7.00017179 7.00008465 ... 6.99535518 6.99733277
6.99879527]]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1,  cmap = "viridis", edgecolor = "none")
plt.show()
```



Ce filtre est un filtre passe haut et si on fait une rotation à ce filre, il va pas changer.

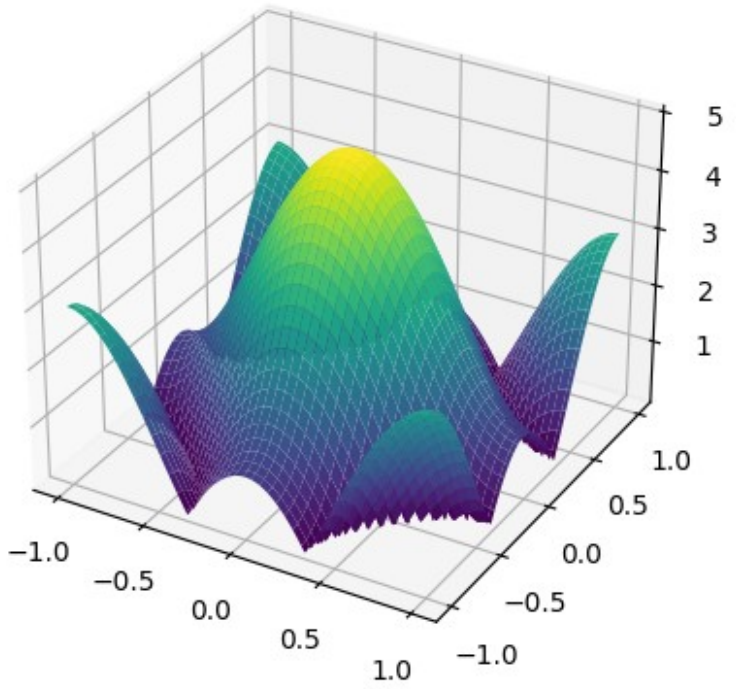
Le type d'application de ce filtre (Filtre de Laplace) est la Détection de contours, le filtre de Laplace est particulièrement efficace pour détecter les contours et les bords dans une image. Il met en évidence les variations rapides, ce qui permet de détecter les transitions abruptes entre les objets et les arrière-plans.

Exercice 03

Question 01

```
M = np.array([[0,1,0],[1,1,1],[0,1,0]])
M1 = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
M2 = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
```

```
x,y,Z1 = freqz(M)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1,  cmap = "viridis")
plt.show()
```




```
io.imshow(img,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc749d8940>
```


```
#Avec l'application du masque sur une image on aura
output = cv2.filter2D(img, cv2.CV_64F, M)
io.imshow(output,cmap="gray")
```

```
/usr/local/lib/python3.10/dist-packages/skimage/io/_plugins/
matplotlib_plugin.py:150: UserWarning: Float image out of standard
range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

```
<matplotlib.image.AxesImage at 0x7cfc7718b4f0>
```

Ce masque est un flou gaussien, il est utilisée pour appliquer un flou doux à une image en moyennant les valeurs des pixels environnants. L'idée est que chaque pixel de l'image de sortie sera la moyenne pondérée des pixels correspondants dans l'image d'entrée. -source internet-

Question 02:

```
D1 = signal.convolve2d(M,M1)
D2 = signal.convolve2d(M,M2)
print("D1= \n",D1)
print("D2= \n",D2)
```

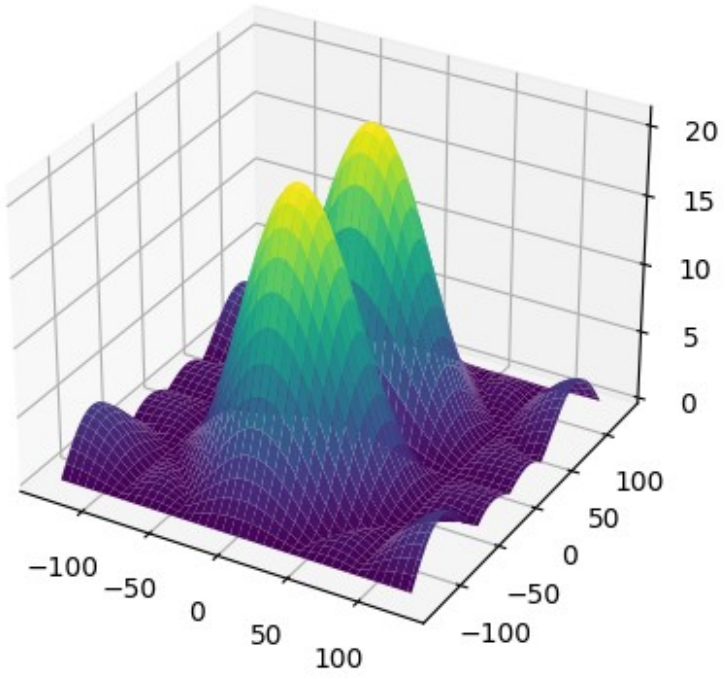
```
D1=
 [[ 0  1  1  1  0]
 [ 1  2  3  2  1]
 [ 0  0  0  0  0]
 [-1 -2 -3 -2 -1]
 [ 0 -1 -1 -1  0]]
D2=
 [[ 0  1  0 -1  0]
 [ 1  2  0 -2 -1]
 [ 1  3  0 -3 -1]
 [ 1  2  0 -2 -1]
 [ 0  1  0 -1  0]]
```

```
print("la taille du masque D1 = ",np.shape(D1))
print("la taille du masque D2 = ",np.shape(D2))
```

```
la taille du masque D1 =  (5, 5)
la taille du masque D2 =  (5, 5)
```

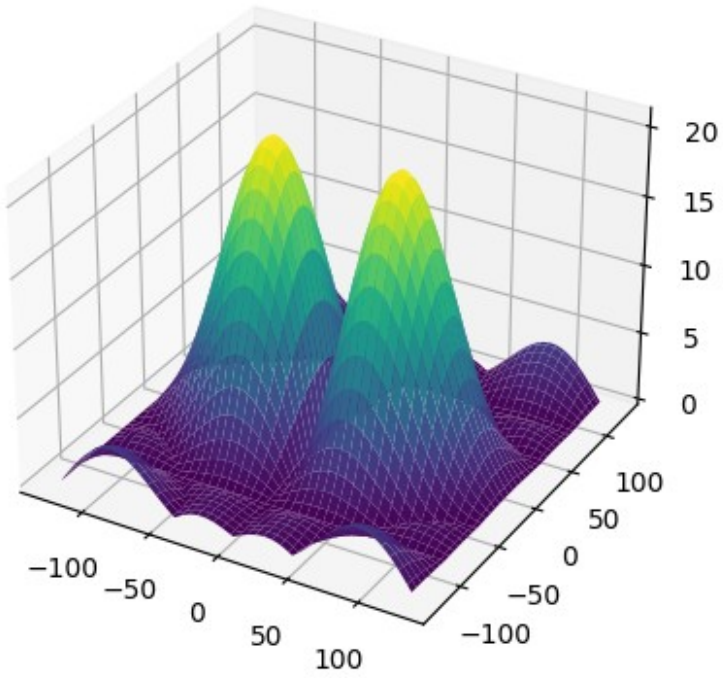
Question 04:

```
img = np.zeros((256,256))
img[126:131,126:131]=D1
a = [i for i in range(-128,128)]
x,y = np.meshgrid(a,a)
Z1 = np.abs(np.fft.fftshift(np.fft.fft2(img)))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis")
plt.show()
```




```
img = np.zeros((256,256))
img[126:131,126:131]=D2
a = [i for i in range(-128,128)]
x,y = np.meshgrid(a,a)
Z2 = np.abs(np.fft.fftshift(np.fft.fft2(img)))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

```
ax.plot_surface(x, y, Z2, cmap = "viridis")
plt.show()
```



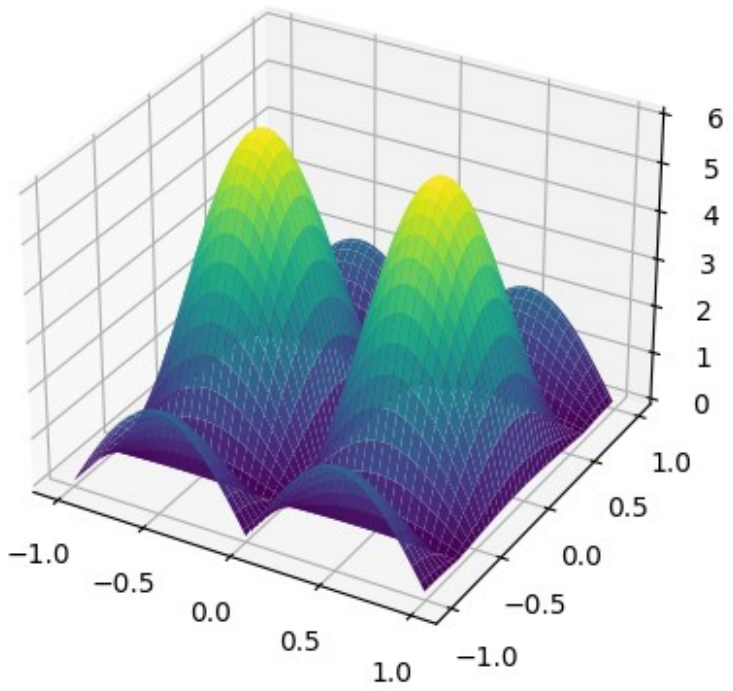
Le type d'application est le filtrage et la detection de contour. Si on compare avec les masques de Prewitt, on trouve que ces filtre on une magnitude plus grande. et ainsi les grandes magnetudes sont condensés autour des deux amplitude maximales. ce qui implique qu ces deux filtre sont presque le meme role que le filtre de Prewitt avec un grand effet de filtrage et plus de lissage par rapport le filtre de Prewitt.

Exercice 04:

```
M0 = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])

x,y,Z0 = freqz(M0)
```

Question 02:

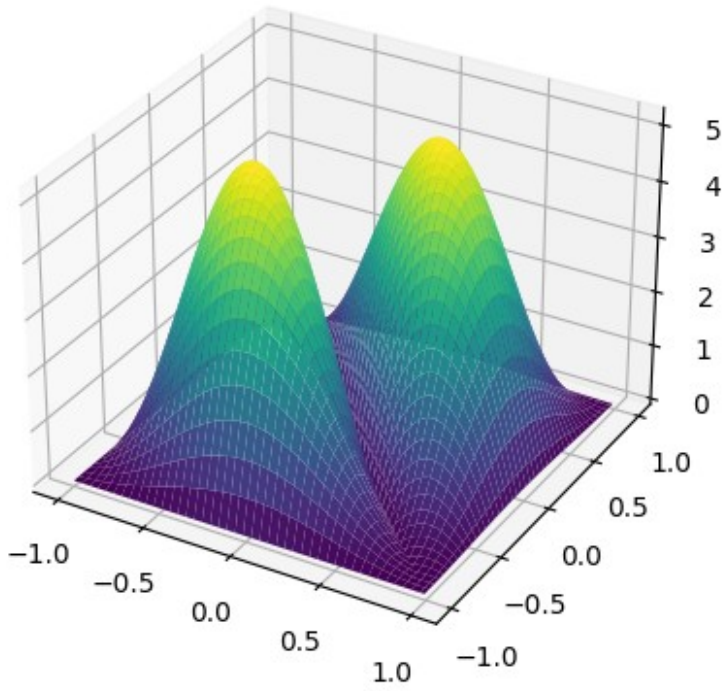
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z0, cmap = "viridis")
plt.show()
```



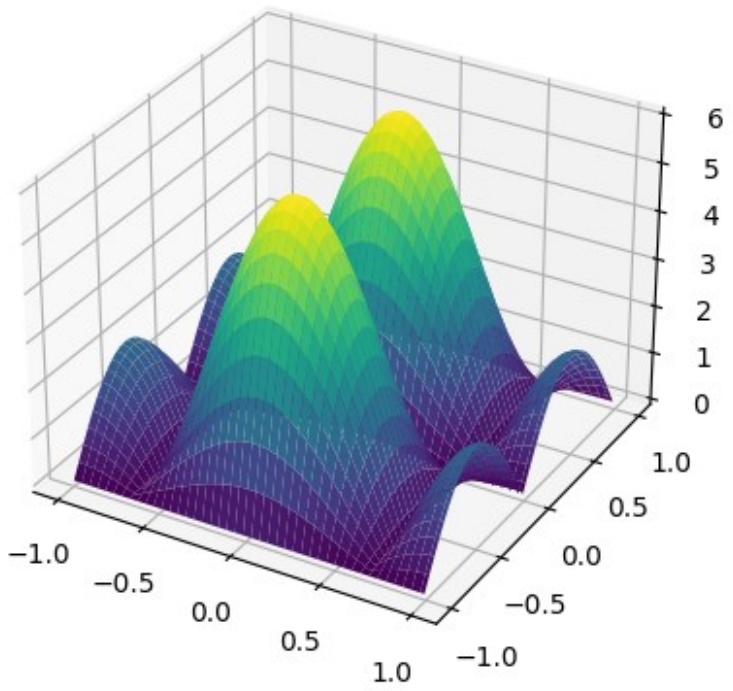
Ce filtre est un filtre de PREWITT il sert a la detection du contour pour les variations horizontales.

Question 03

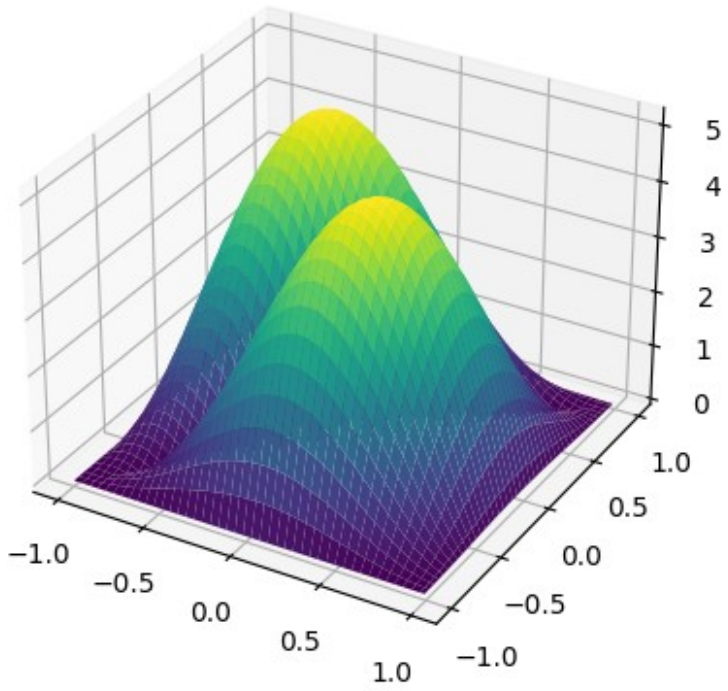
```
M1 = np.array([[1,1,0],[1,0,-1],[0,-1,-1]])
x,y,Z1 = freqz(M1)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z1, cmap = "viridis")
plt.show()
```



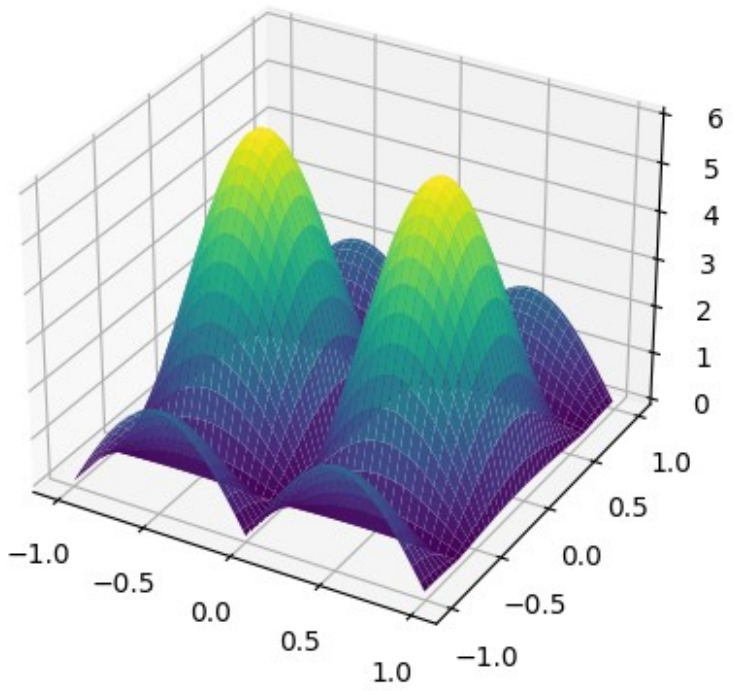

```
M2 = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
x,y,Z2 = freqz(M2)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z2, cmap = "viridis")
plt.show()
```



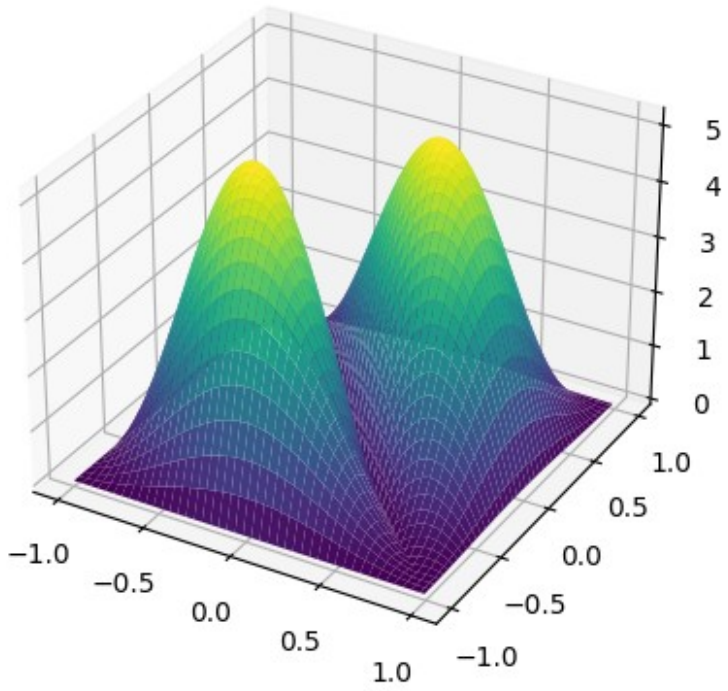

```
M3 = np.array([[0,-1,-1],[1,0,-1],[1,1,0]])
x,y,Z3 = freqz(M3)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z3, cmap = "viridis")
plt.show()
```



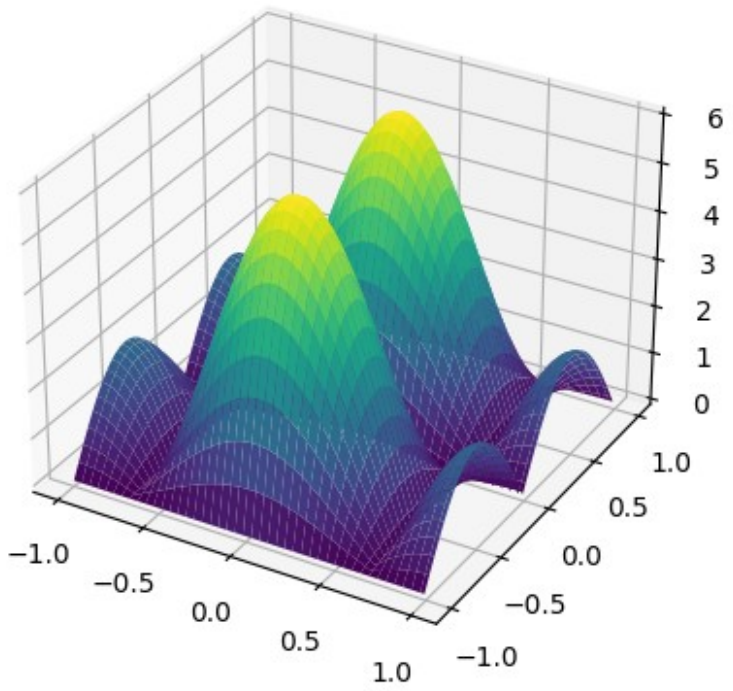

```
M4 = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
x,y,Z4 = freqz(M4)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z4, cmap = "viridis")
plt.show()
```



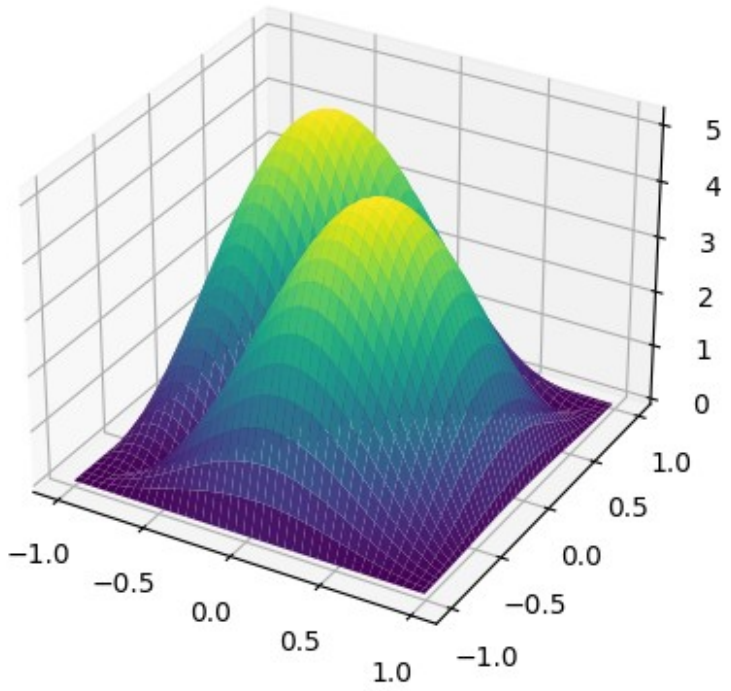

```
M5 = np.array([[-1,-1,0],[-1,0,1],[0,1,1]])
x,y,Z5 = freqz(M5)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z5, cmap = "viridis")
plt.show()
```




```
M6 = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
x,y,Z6 = freqz(M6)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z6, cmap = "viridis")
plt.show()
```

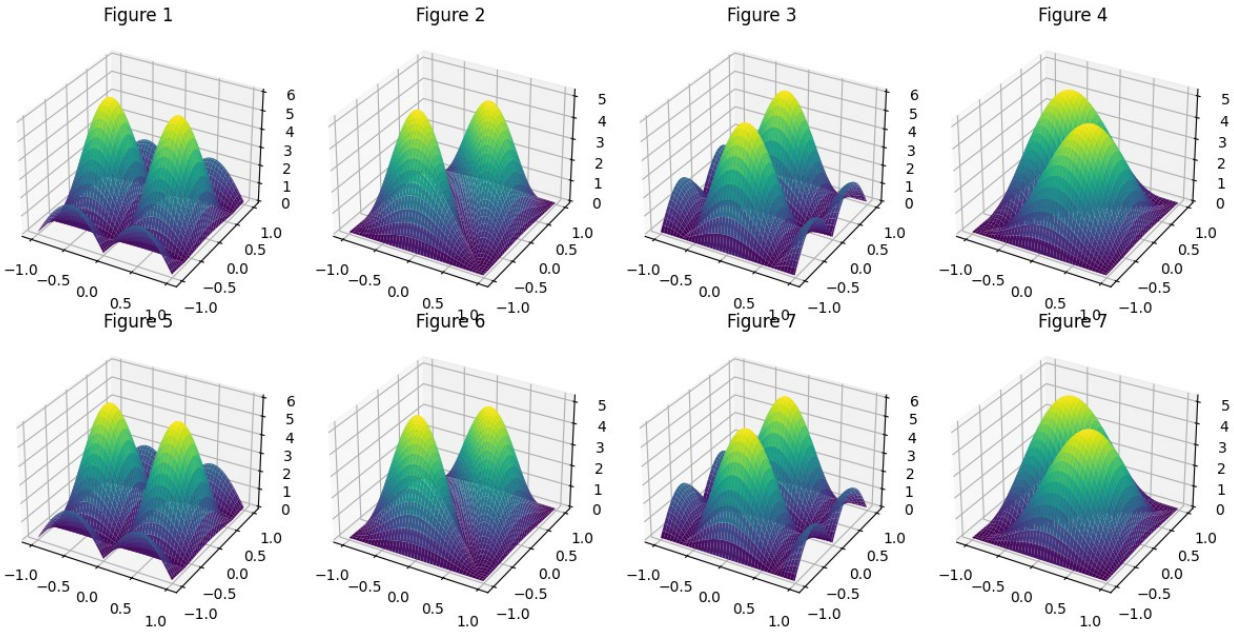



```
M7 = np.array([[0,1,1],[-1,0,1],[-1,-1,0]])
x,y,Z7 = freqz(M7)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, Z7, cmap = "viridis")
plt.show()
```




```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Create a figure with 2 rows and 4 columns of subplots
fig, axes = plt.subplots(2, 4, figsize=(12, 6),
subplot_kw={'projection': '3d'})

# Plot data on each subplot
axes[0, 0].plot_surface(x, y, Z0, cmap="viridis")
axes[0, 0].set_title("Figure 1")

axes[0, 1].plot_surface(x, y, Z1, cmap="viridis")
axes[0, 1].set_title("Figure 2")

axes[0, 2].plot_surface(x, y, Z2, cmap="viridis")
axes[0, 2].set_title("Figure 3")

axes[0, 3].plot_surface(x, y, Z3, cmap="viridis")
axes[0, 3].set_title("Figure 4")

axes[1, 0].plot_surface(x, y, Z4, cmap="viridis")
axes[1, 0].set_title("Figure 5")

axes[1, 1].plot_surface(x, y, Z5, cmap="viridis")
axes[1, 1].set_title("Figure 6")

axes[1, 2].plot_surface(x, y, Z6, cmap="viridis")
```

```
axes[1, 2].set_title("Figure 7")

axes[1, 3].plot_surface(x, y, Z7, cmap="viridis")
axes[1, 3].set_title("Figure 7")

# Adjust spacing between subplots
plt.tight_layout()

# Show the combined figure with all seven subplots
plt.show()
```



Ces filtres sont particulièrement utiles pour la détection de contours dans une image mais avec differents rotation (directions)

L'appellation "opérateurs compas" peut être justifiée en raison de la similitude de ces filtres avec les points cardinaux sur une boussole. Les filtres de Prewitt sont conçus pour détecter les contours dans quatre directions principales : nord, sud, est et ouest, ce qui rappelle les directions cardinales d'une boussole. Les filtres de Prewitt avec différentes rotations sont essentiellement des variantes de ces filtres de base, adaptées pour détecter les contours dans des directions spécifiques, tout comme un compas peut pointer dans différentes directions. Cette analogie avec un compas aide à comprendre intuitivement la fonction des filtres de Prewitt et leurs rotations dans le contexte du traitement d'image.

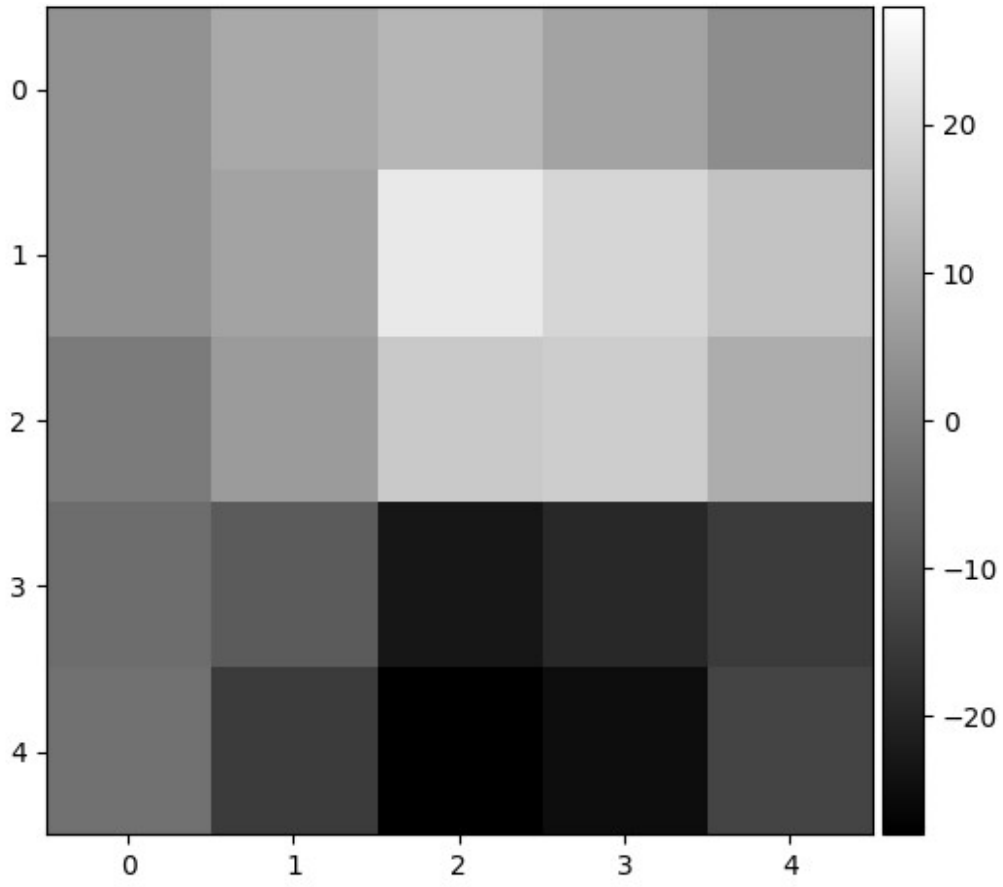
Question 04:

```
I = np.array([[4,5,3],[4,4,15],[3,12,13]])

I
```

```
array([[ 4,  5,  3],
       [ 4,  4, 15],
       [ 3, 12, 13]])
```

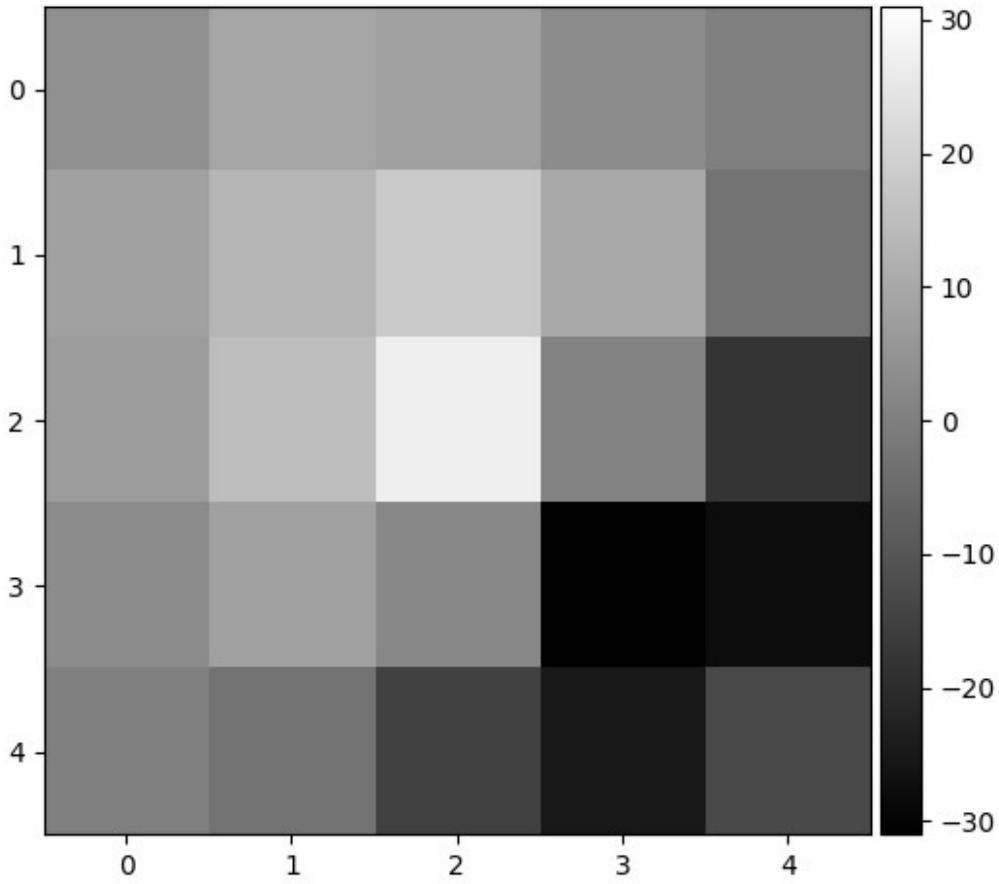
```
#L'application du masque 0
output = signal.convolve2d(I, M0)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7a689930>
```



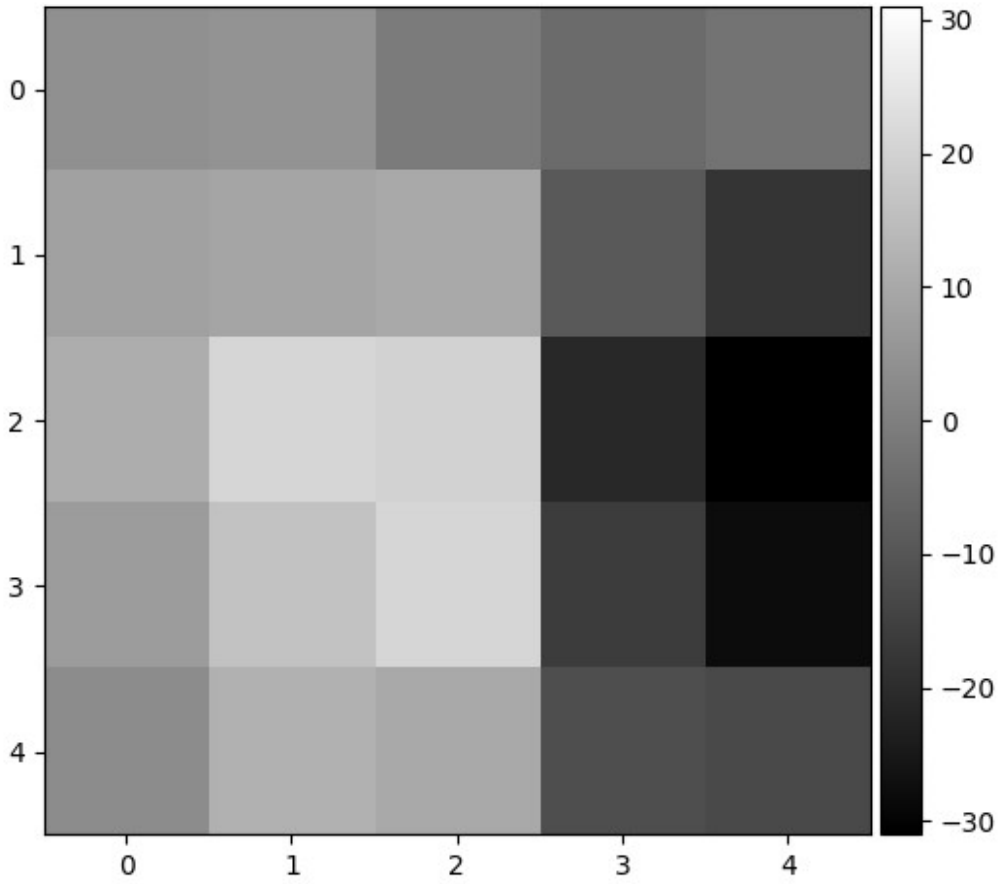

```
#L'application du masque 1
output = signal.convolve2d(I, M1)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7b091d80>
```



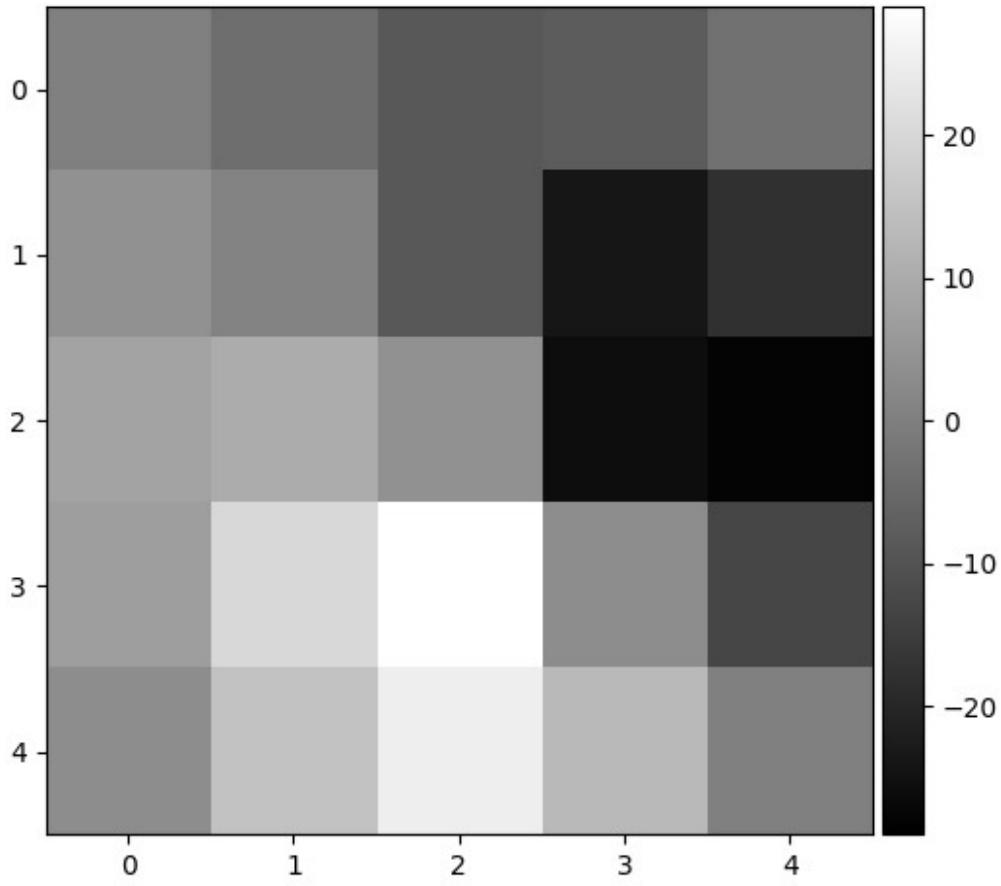

```
#L'application du masque 2
output = signal.convolve2d(I, M2)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7f7688e0>
```



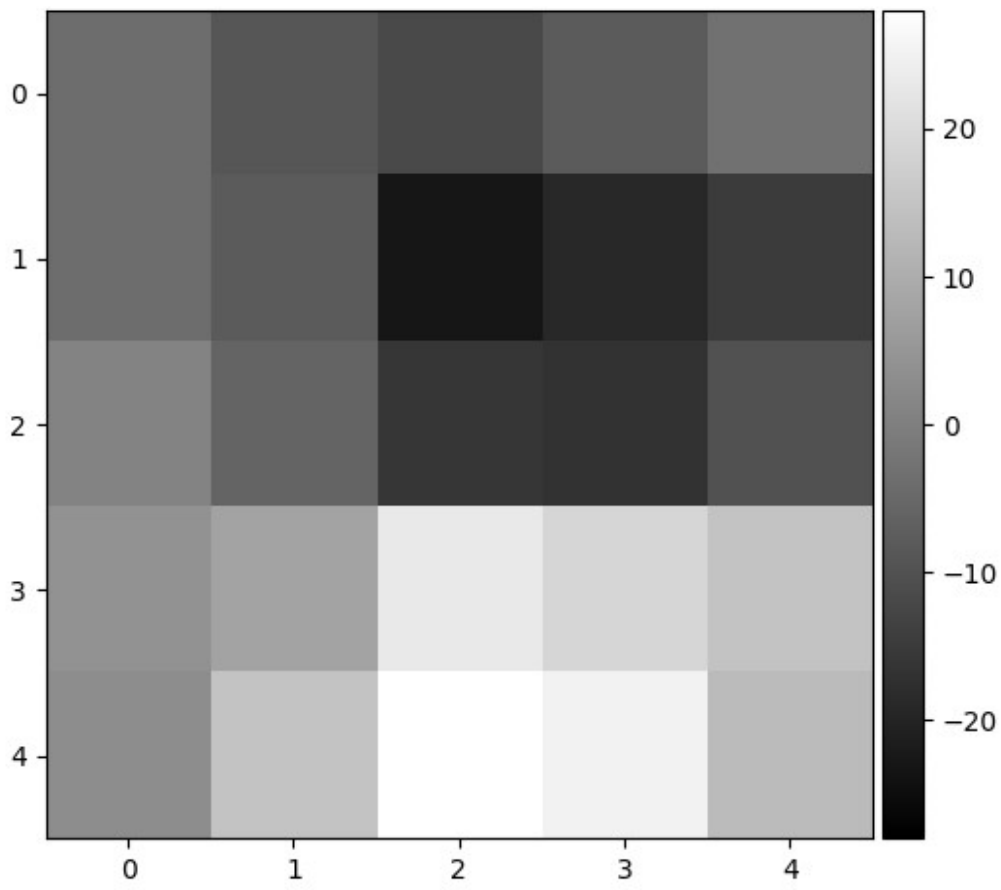

```
#L'application du masque 3
output = signal.convolve2d(I, M3)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc7464a170>
```



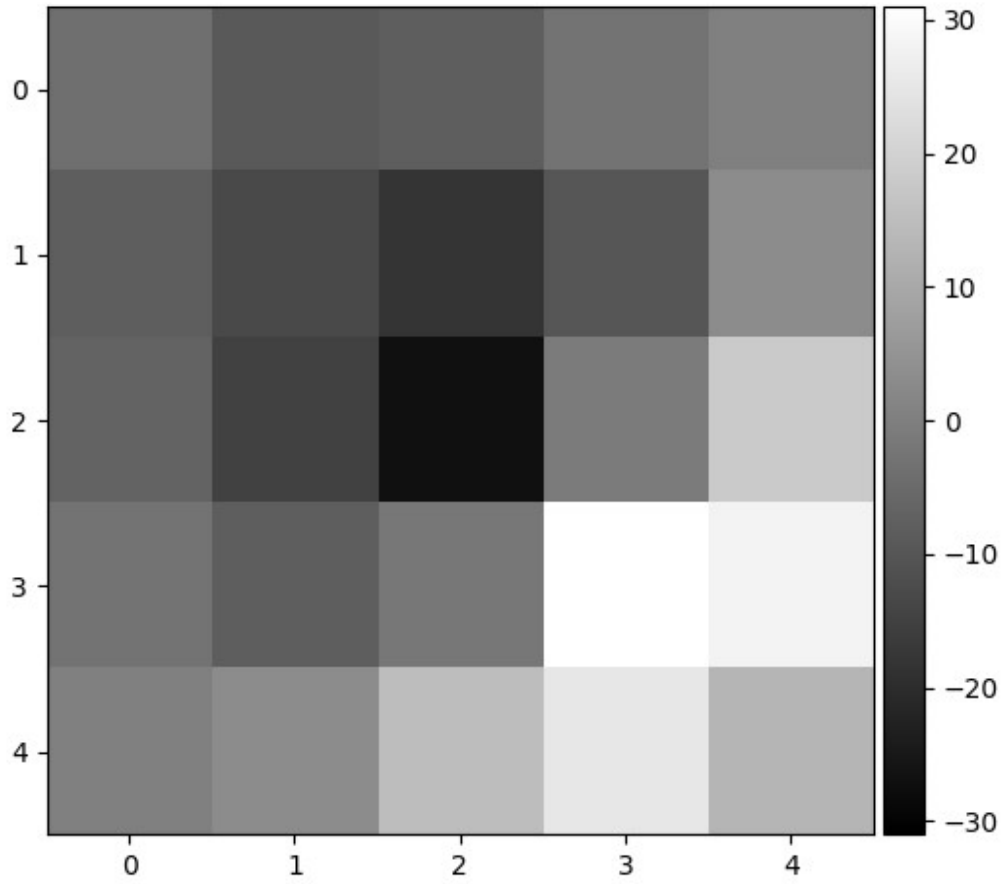

```
#L'application du masque 4
output = signal.convolve2d(I, M4)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc736b87f0>
```



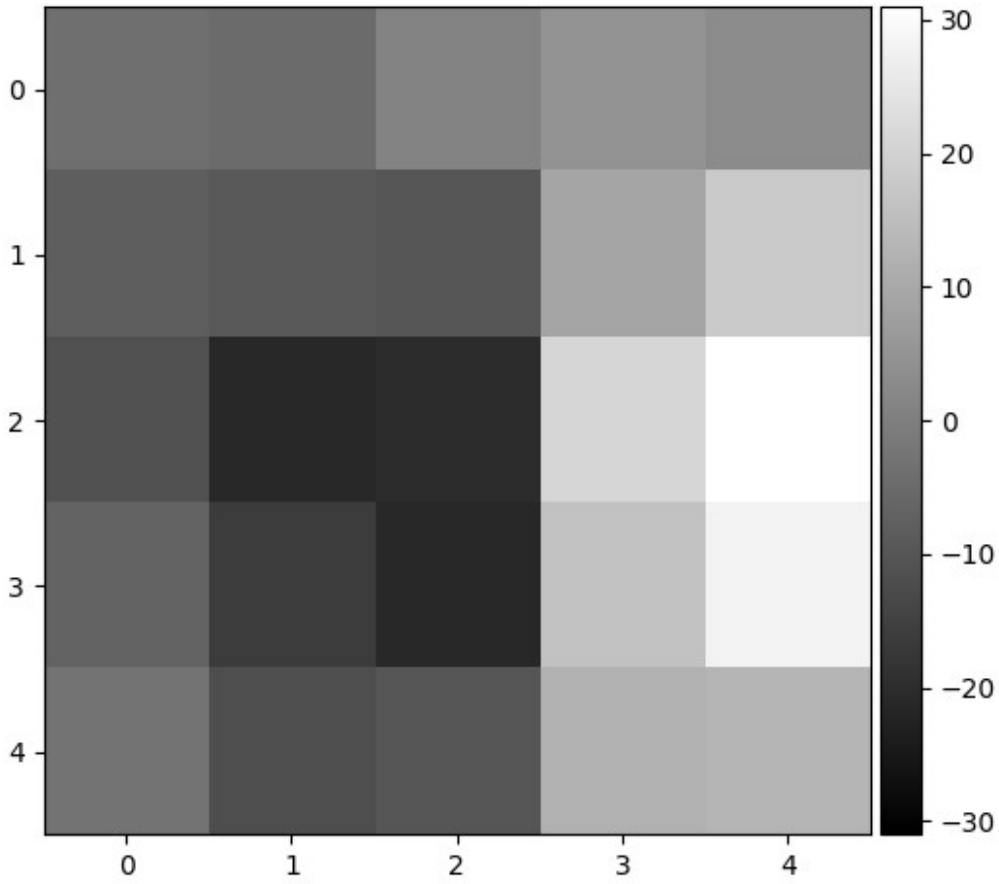

```
#L'application du masque 5
output = signal.convolve2d(I, M5)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc77cd40d0>
```



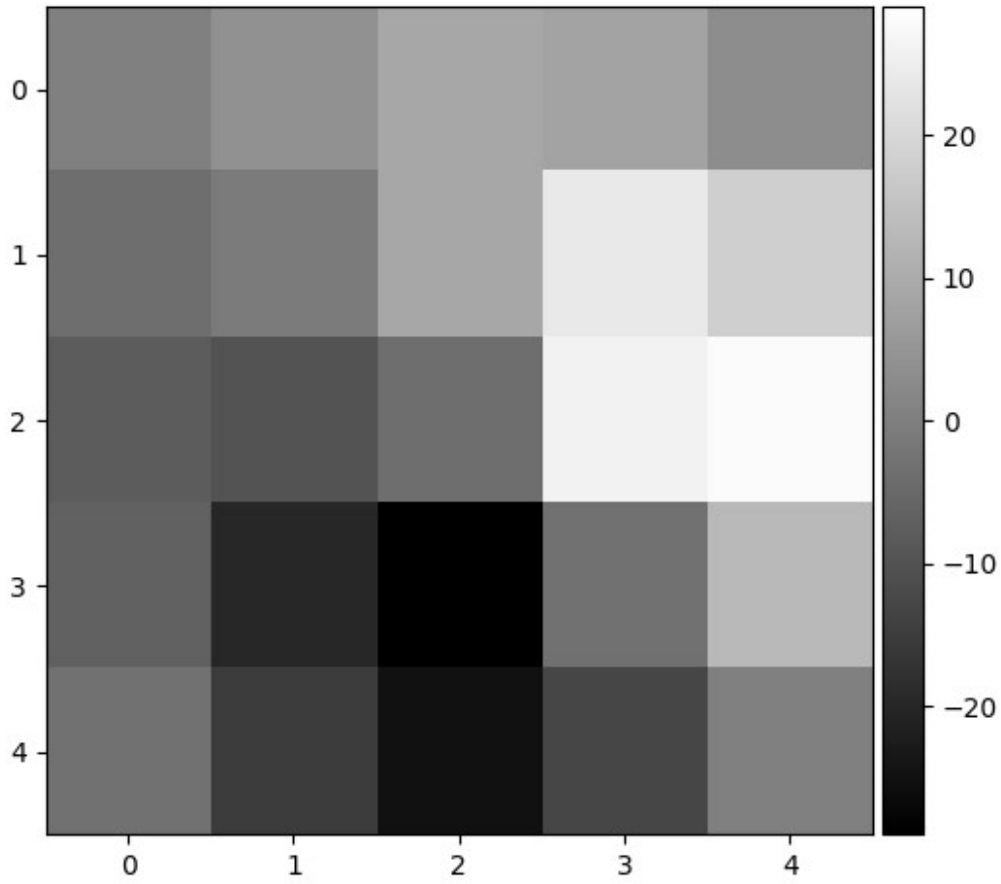

```
#L'application du masque 6
output = signal.convolve2d(I, M6)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc74163fa0>
```




```
#L'application du masque 7
output = signal.convolve2d(I, M7)
io.imshow(output,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cfc74775d50>
```



A partir de ce qu'on a eu, on peut voire que le masque 1 avec une direction de -45 degrés (suivant la direction de l'orientation de l'horloge) detecte bien le contour au point (0,0).

Question 05: Il faut garder les huit masques? ça dépends l'application, si on veut seulement detecter le contour au point de (0,0) on peut seulement utiliser le masque 01. mais pour d'autres contours, les autres masques peuvent etre utilisable.

Exercice 02 :

1- La réponse impulsionnelle du filtre :

$$h(x,y) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2- La réponse fréquentielle de $h(x,y)$:

$$TF\big(h(x,y)\big) = TF\left(\begin{bmatrix} 0 & 1 & 1 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right) = e^{j2\pi u} + e^{j2\pi v} - 4 + e^{-j2\pi u} + e^{-j2\pi v}$$

$$\boldsymbol{TF\big(h(x,y)\big) = 2\cos(2\pi u) + 2\cos(2\pi v) - 4}$$

Exercice 03 :

3- Les masques D1 et D2 :

On commence par pivoter les filtre M1 et M2 et après on fais le produit de convolution :

$$M1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$M1' = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$M2' = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Par produit de convolution on trouve :

$$D1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & -3 & -2 & -1 \\ 0 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$D2 = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 \\ 1 & 2 & 0 & -2 & -1 \\ 1 & 3 & 0 & -3 & -1 \\ 1 & 2 & 0 & -2 & -1 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

Le calcule est vérifié par la fonction signal.convolve2d.