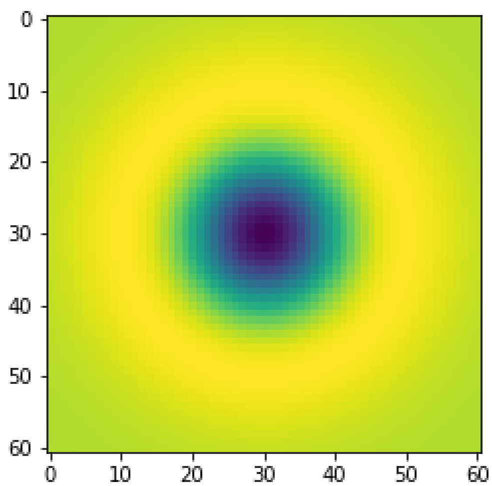
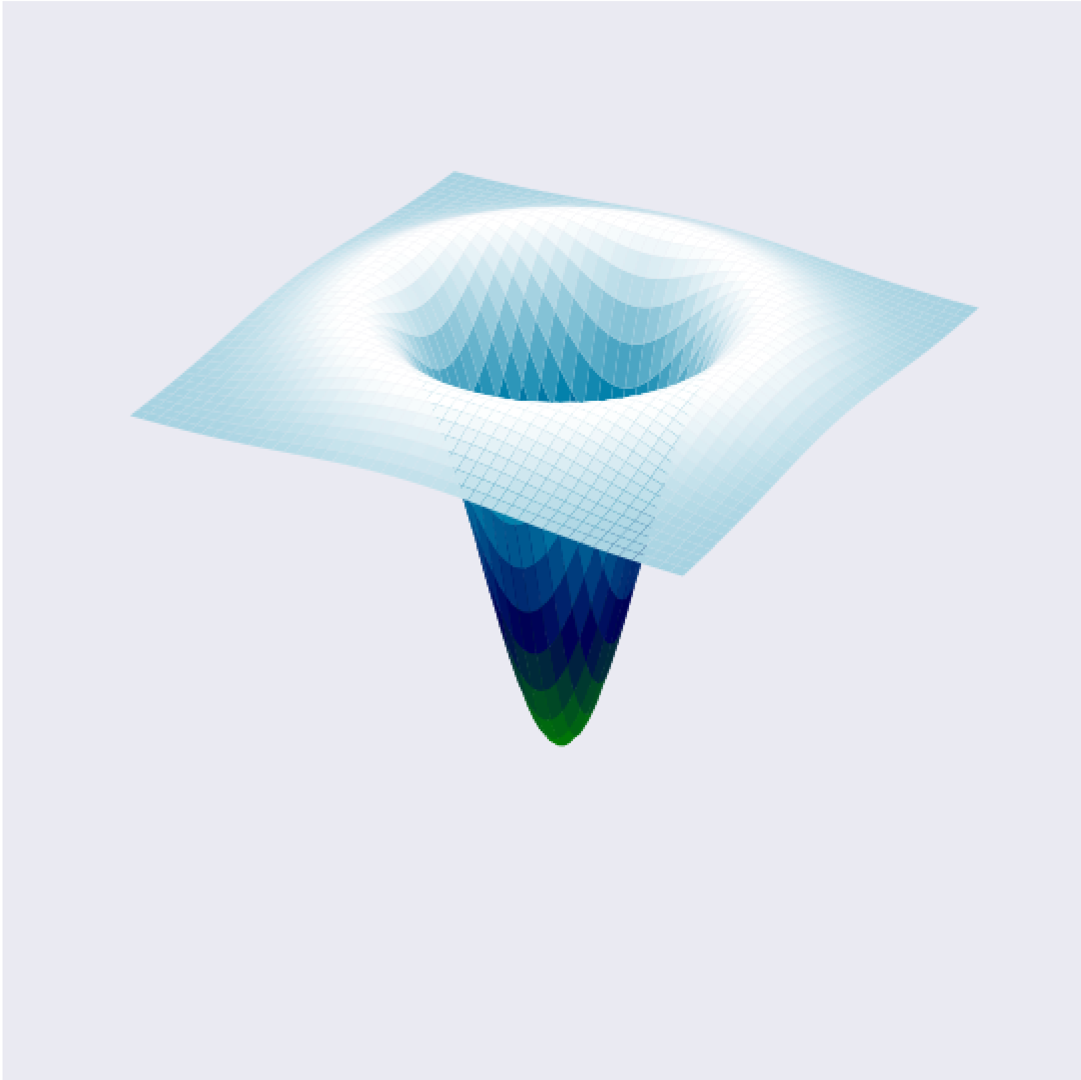
190432J

Pathirana R.P.U.A.

In [ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

sigma = 10
hw = 3 *sigma
X, Y = np.meshgrid(np.arange(-hw,hw+1,1),np.arange(-hw,hw+1,1))

log = 1/(2*np.pi*sigma**2)*(X**2/(sigma**2) + Y**2/(sigma**2) - 2)*np.exp(-(X**2 + Y**


plt.imshow(log)
plt.show()
```

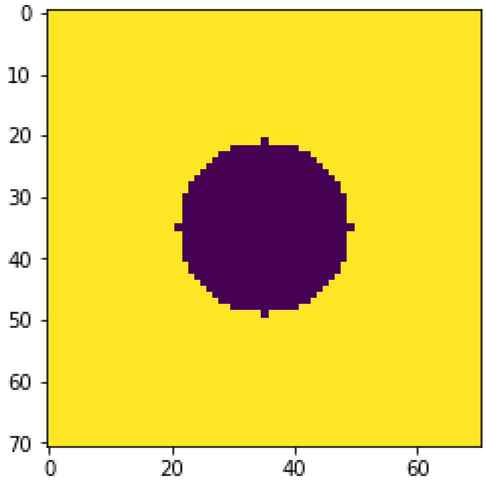


In [ ]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111,projection = '3d')

surf = ax.plot_surface(X,Y,log,cmap=cm.ocean,linewidth=0,antialiased = True)

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
plt.axis('off')
plt.show()
```



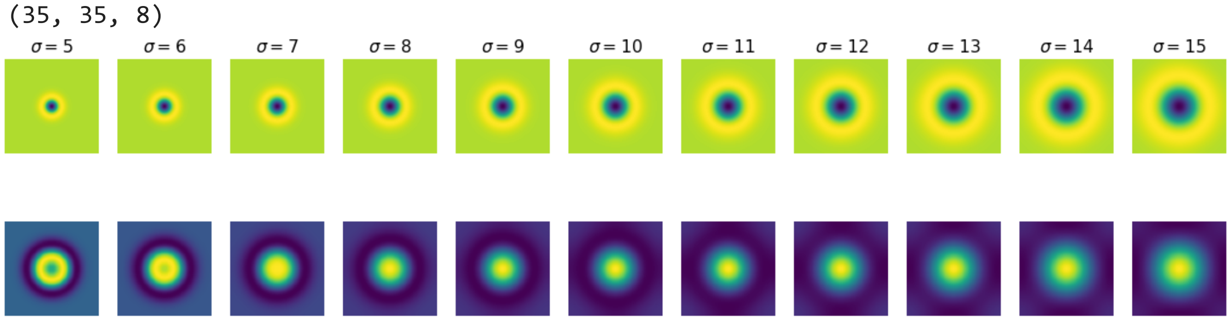
In [ ]:

```
#Q2

import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
w,h = 71,71
hw,hh = w//2,h//2

f = np.ones((h,w),dtype=np.float32)*255
X, Y = np.meshgrid(np.arange(-hh,hh+1,1),np.arange(-hw,hw+1,1))

r = w//5 #4
f *= X**2 +Y**2 > r**2

plt.imshow(f)
plt.show()
```



In [ ]:
```
s = 11
fig, ax = plt.subplots(2,s, figsize= (20,5))
scale_space = np.empty((h,w,s),dtype=np.float32)
sigmas = np.arange(5,16,1)
for i,sigma in enumerate(sigmas):
    log_hw = 3*np.max(sigmas)
    X, Y = np.meshgrid(np.arange(-log_hw,log_hw+1,1),np.arange(-log_hw,log_hw+1,1))
    log = 1/(2*np.pi*sigma*2)*(X**2/(sigma**2) + Y**2/(sigma**2) - 2)*np.exp(-(X**2 +
    f_log = cv.filter2D(f,-1,log)
    scale_space [:,:,i] = f_log
    ax[0,i].imshow(log)
    ax[0,i].axis('off')
    ax[0,i].set_title('$\sigma = {}$'.format(sigma))
    ax[0,i].imshow(log)
    ax[1,i].imshow(f_log)
    ax[1,i].axis('off')

indices = np.unravel_index(np.argmax(scale_space,axis=None),scale_space.shape)
print(indices) # r = \sqrt{2}*sigma
plt.style.use('seaborn')
```

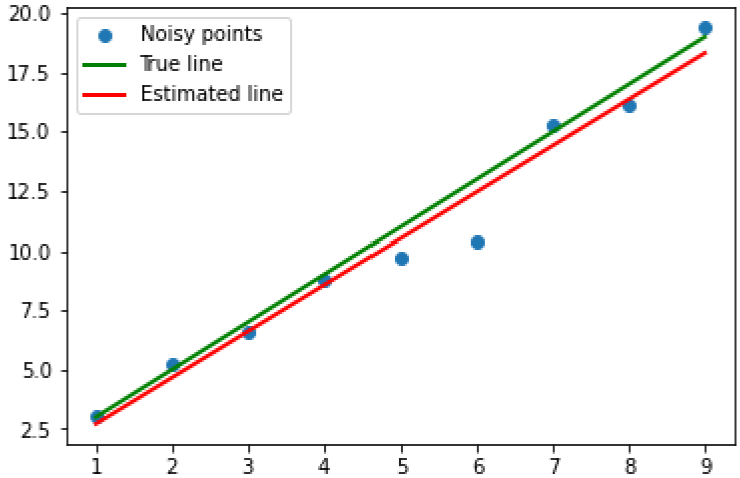
```
(35, 35, 8)
```



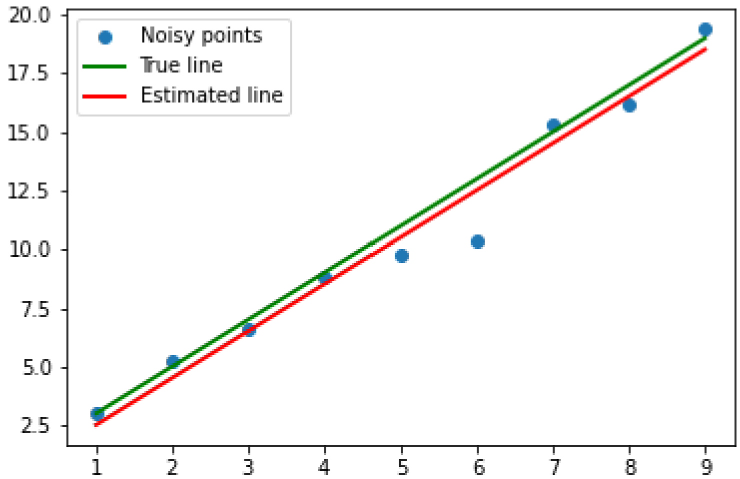
In [ ]:
```
m = 2 # Line equation : y = m*x + c . m is the slope .c is the intercept.

c = 1
x = np.arange (1 ,10 , 1)
np.random.seed(45)
noise = np.random.randn( len(x) )
o = np.zeros(x.shape)
# o [-1] = 20
y = m*x + c + noise + o

n = len(x)
```

```python
X = np.concatenate([x.reshape(n,1),np.ones((n,1))],axis=1)
B = np.linalg.pinv(X.T @ X) @ X.T @ y
mstar = B[0]
cstar = B[1]

plt.plot(x,y,'o',label='Noisy points')
plt.plot([x[0],x[-1]],[m*x[0]+c,m*x[-1] + c], color='g',linewidth=2, label=r'True line
plt.plot([x[0],x[-1]],[mstar*x[0] + cstar,mstar*x[-1] + cstar], color='r', linewidth=2
plt.legend()
```

Out[ ]: `<matplotlib.legend.Legend at 0x19331d67f40>`



In [ ]:
```python
m = 2 # Line equation : y = m*x + c . m is the slope .c is the intercept.
c = 1
x = np.arange (1 ,10 , 1)
np.random.seed(45)
noise = np.random.randn( len(x) )
o = np.zeros(x.shape)
# o [-1] = 20
y = m*x + c + noise + o

n = len(x)

u11 = np.sum((x - np.mean(x))**2)
u12 = np.sum((x - np.mean(x))*(y - np.mean(y)))
u21 = u12
u22 = np.sum((y - np.mean(y))**2)

U = np.array([[u11,u12],[u21,u22]])
W,V = np.linalg.eig(U)
ev_corresponding_to_smallest_ev = V[:,np.argmin(W)]

a = ev_corresponding_to_smallest_ev[0]
b = ev_corresponding_to_smallest_ev[1]
d = a*np.mean(x) + b*np.mean(y)

mstar = -a/b
cstar = d/b

plt.plot(x,y,'o',label='Noisy points')
plt.plot([x[0],x[-1]],[m*x[0]+c,m*x[-1] + c], color='g',linewidth=2, label=r'True line
plt.plot([x[0],x[-1]],[mstar*x[0] + cstar,mstar*x[-1] + cstar], color='r', linewidth=2
plt.legend(loc = 'best')
plt.show()
```


Noisy points
True line
Estimated line
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1 2 3 4 5 6 7 8 9

In [ ]: