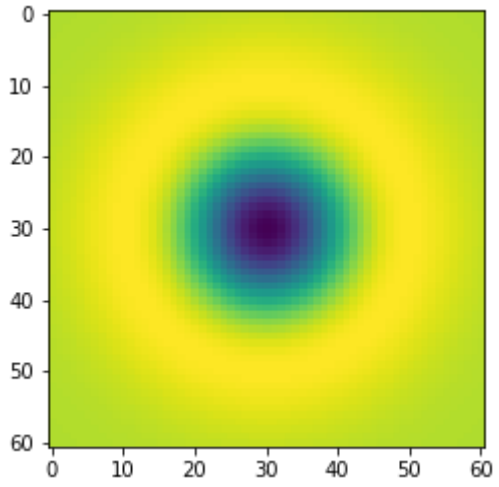
Name: C. J. Kurukulasuriya

Index No.: 190337X

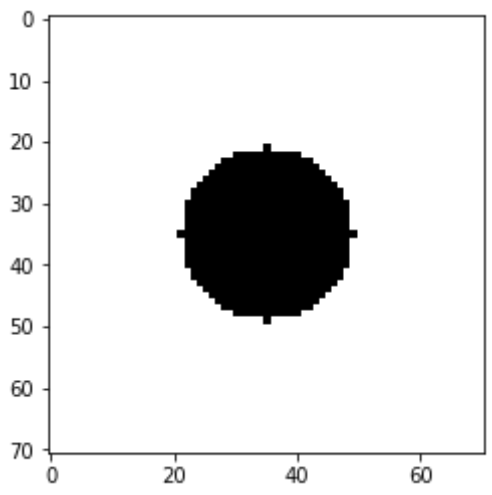
Q1

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

In [ ]:
```
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**2)/(2*sigma**2))

# fig = plt.figure(figsize=(10,10))
# surf = ax.plot_surface(X, Y, log, cmap)

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



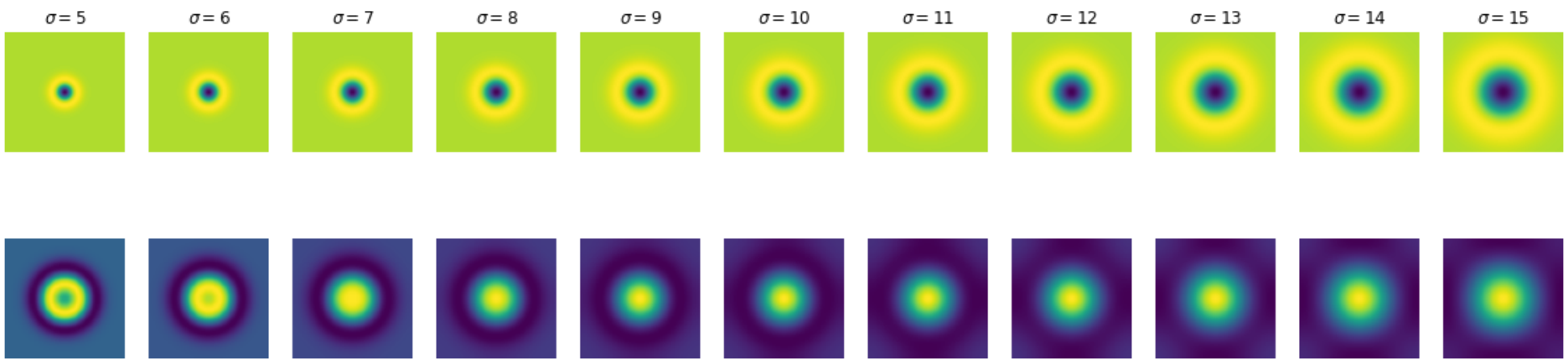
In [ ]:
```
w, h = 71, 71
hw, hh = w//2, h//2

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 #14
f *= X**2+ Y**2 > r**2

plt.imshow(f, cmap = "gray")
plt.show()
```



In [ ]:
```
s = 11
fig, ax = plt.subplots(2, s, figsize = (20,5))
sigmas = np.arange(5,16,1)
scale_space = np.empty((h,w,s), dtype=np.float32)
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+Y**2)/(2*sigma**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[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) #14/root(2)
```

```
print(sigmas[indices[2]])
plt.show()
```

```
(35, 35, 5)
10
```



In [ ]:

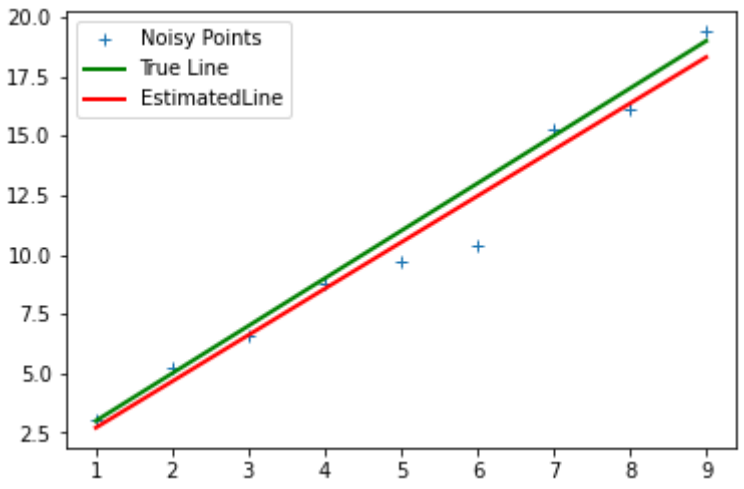
```
img1 = cv.imread('Images/img1.ppm')
img2 = cv.imread('Images/img2.ppm')

cv.imshow('img1', img1)
cv.waitKey(0)
cv.imshow('img2', img2)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
img1 = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
img2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)

sift = cv.SIFT_create()

keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2, None)

bf = cv.BFMatcher(cv.NORM_L1, crossCheck = True)

matches = bf.match(descriptors_1, descriptors_2)
matches = sorted(matches, key = lambda x:x.distance)

fig, ax = plt. subplots(figsize=(10,10))
ax.axis('off')
img3 = cv.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:50], img2, flags = 2)
plt.imshow(img3)
plt.show()
```

Q2

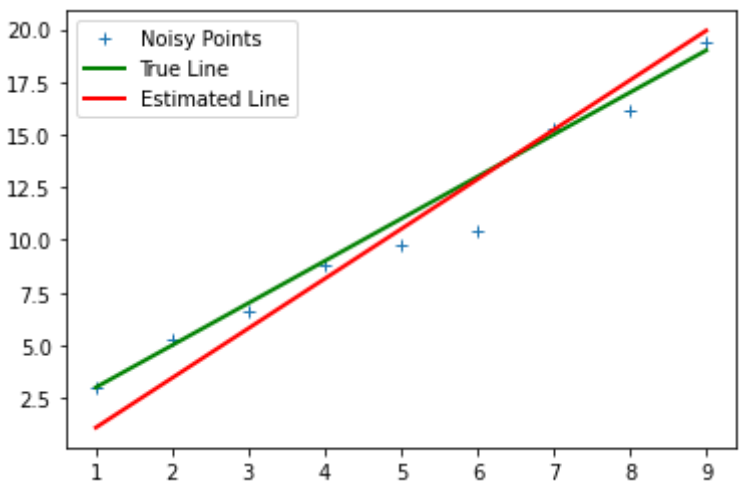
In [ ]:

```
m = 2
c = 1

x = np.arange(1,10,1)
np.random.seed(45)
sigma = 1
n = sigma*np.random.randn(len(x))
o = np.zeros(x.shape)
y = m*x + c +n + o

n = len(x)
X = np.concatenate([x.reshape(n,1), np.ones((n,1))], axis = 1)
B = np.linalg.pinv(X.T @ X) @X.T @y

m_star = B[0]
c_star = B[1]

plt.plot(x,y, '+',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]], [m_star*x[0]+c_star, m_star*x[-1] +c_star], color = 'r', linewidth = 2, label = r"EstimatedLine")
plt.legend()
plt.show()
```



In [ ]:

```
m = 2
c = 1
x = np.arange(1,10,1)

np.random.seed(45)
sigma = 1
n = sigma*np.random.randn(len(x))
o = np.zeros(x.shape)
y = m*x + c +n + o

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, u22], [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)

m_star = -a/b
c_star = d/b

plt.plot(x, y, '+', label = r'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]], [m_star*x[0]+c_star, m_star*x[-1] +c_star], color = 'r', linewidth = 2, label = r"Estimated Line")
plt.legend()
plt.plot()
plt.show()
```



In [ ]: