```
from scipy import real
from scipy.sparse.linalg import spsolve
import math
# load modules
import matplotlib.pyplot as plt

from skimage.color import rgb2gray
from skimage import data
from skimage.transform import resize

# Numpy is useful for handling arrays and dense matrices (a matrix with a lot of
nonzeros).
import numpy as np
import scipy
from scipy.sparse import coo_matrix
from scipy.sparse import csc_matrix
from scipy.sparse import csr_matrix

from scipy.sparse import diags
from scipy.sparse import kron
from scipy.sparse import identity

# load image
img = data.stereo_motorcycle()[0]
img = rgb2gray(img)*255 # convert to gray and change scale from (0,1) to (0,255).

m = img.shape[0] # ROWS
n = img.shape[1] # COLS

plt.figure(1, figsize=(10, 10))
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
# plt.show()


##############################
# Question 1
##############################
# difference operator
diagsOfI = [1 for i in range(min(n, m))]
# print(diagsOfI)
I = diags([1], [0],shape=(n,n),dtype='int8').toarray()
# print('\nMatrix I -----')
# print(I)
J = diags([-1,1],[0,1],shape=(m,m),dtype='int8').toarray()
# print('\nMatrix j -----')
# print(J)
Dh = kron(J, I, format="csr")
Dv = kron(I, J, format="csr")
# print(D)


x = csr_matrix(np.reshape(img,(n*m,1)))

Dh_x = Dh*x.toarray()
Dh_x = Dh_x.reshape(m,n)
# vertical Difference Operator: Done
Dv_x = Dv*x.toarray()
Dv_x = Dv_x.reshape(m,n)


##############################
# Question 2
```

```
##############################
# show figures
plt.figure(2, figsize=(10, 10))
plt.title('applied horizontal difference operator')
plt.imshow(Dh_x, cmap='gray', vmin=0, vmax=255)
# plt.show()

plt.figure(3, figsize=(10, 10))
plt.title('applied vertical difference operator')
plt.imshow(Dv_x, cmap='gray', vmin=0, vmax=255)
# plt.show()


##############################
# Question 2 add noise to img
##############################
mean_ = 0
standard_deviation = 30
dimensions = (m,n)

noise = np.random.normal(mean_,standard_deviation,dimensions)

noisy_image = img + noise

plt.figure(4, figsize=(10, 10))
plt.title('added noise to image')
plt.imshow(noisy_image, cmap='gray', vmin=0, vmax=255)

##############################
# Quetion 3: Denoise the image.
##############################
from scipy.sparse.linalg import spsolve
# This method can be used to solve linear systems.
from scipy import real
# This module can be used to convert arrays and matrices from complex to real.

# Use the following lambda list to tune the denoising problem.
lambda_list = [0.1, 3, 10]

# convert noisy_image to shape=(n*m, 1) matrix
z_noisy = csr_matrix(np.reshape(noisy_image, (n*m, 1)))
I_mn_mn = identity(m*n)
count = 4
for lambda_ in lambda_list:
    count +=1
    print(lambda_)
    LHS = lambda_*real(Dh.transpose().dot(Dh) + Dv.transpose().dot(Dv)) + I_mn_mn
    new_img = spsolve(LHS,z_noisy).reshape(m,n)
    # print(LHS.shape)
    plt.figure(count, figsize=(10, 10))
    plt.title('lambda = ' + str(lambda_))
    plt.imshow(new_img, cmap='gray', vmin=0, vmax=255)
plt.show()
```