ECE 637: Lab 2

Rahul Deshmukh

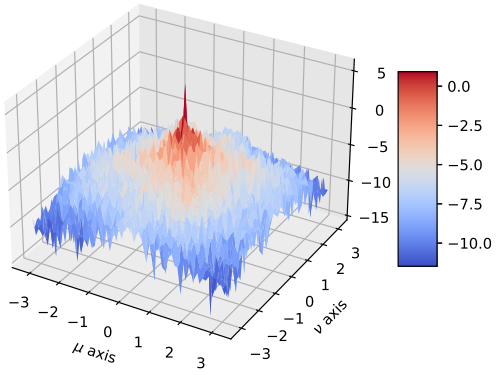
February 3, 2021

Section 1 Report

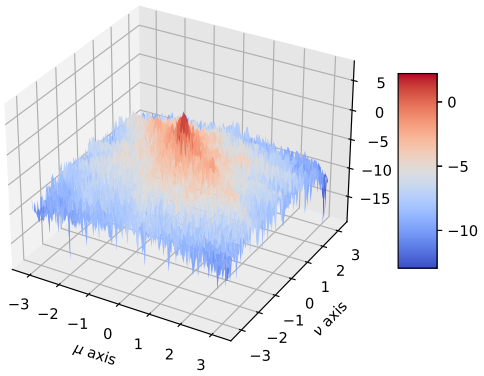
1. Gray scale image img04g.tif

Figure 1: Input gray scale image

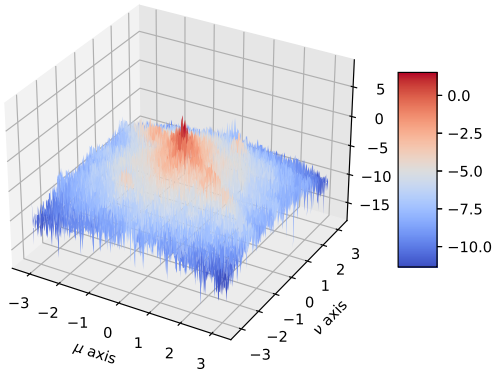
2. Power Spectral density plots:



(a) PSD for 64x64



(b) PSD for 128x128



(c) PSD for 256x256

3. Improved PSD using *BetterSpecAnal(x)* function:

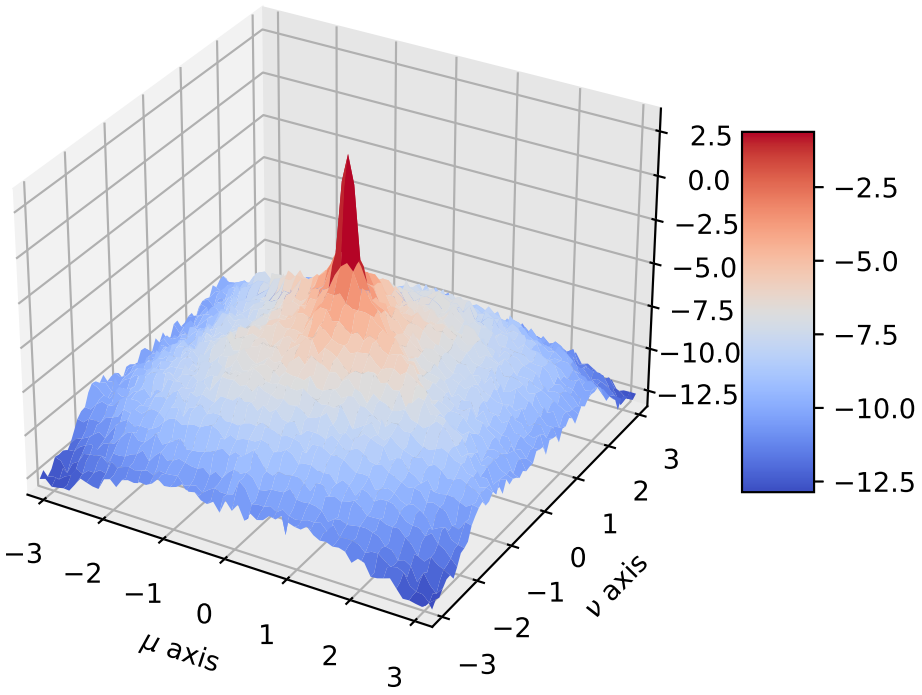


Figure 3: improved PSD for img04g.tif

4. For python code of *BetterSpecAnal(x)* function refer to Listing 1 at page 8.

Section 2 Report

1. The image $255 * (x + 0.5)$:

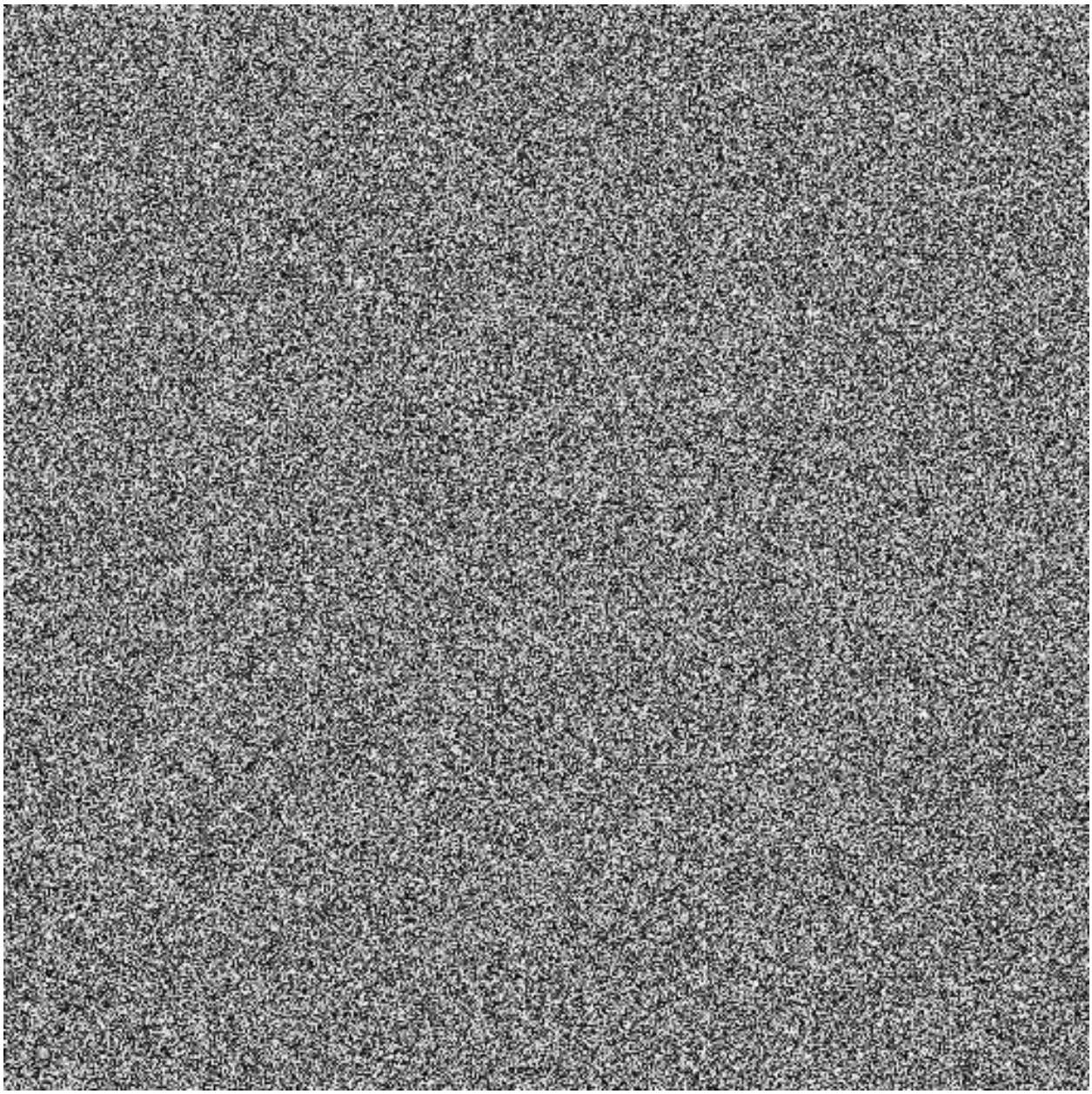


Figure 4: random gray scale image generated

We need to find the difference equation for the given transfer function:

$$H(z_1, z_2) = \frac{Y(z_1, z_2)}{X(z_1, z_2)} = \frac{3}{1 - 0.99z_1^{-1} - 0.99z_2^{-1} + 0.981z_1^{-1}z_2^{-1}}$$

$$\Rightarrow Y(z_1, z_2) = 3X(z_1, z_2) + 0.99(z_1^{-1} + z_2^{-1})Y(z_1, z_2) - 0.981z_1^{-1}z_2^{-1}Y(z_1, z_2)$$

Therefore the difference equation is given by:

$$y(m, n) = 3x(m, n) + 0.99(y(m - 1, n) + y(m, n - 1)) - 0.981y(m - 1, n - 1)$$

2. The image $y + 127$:

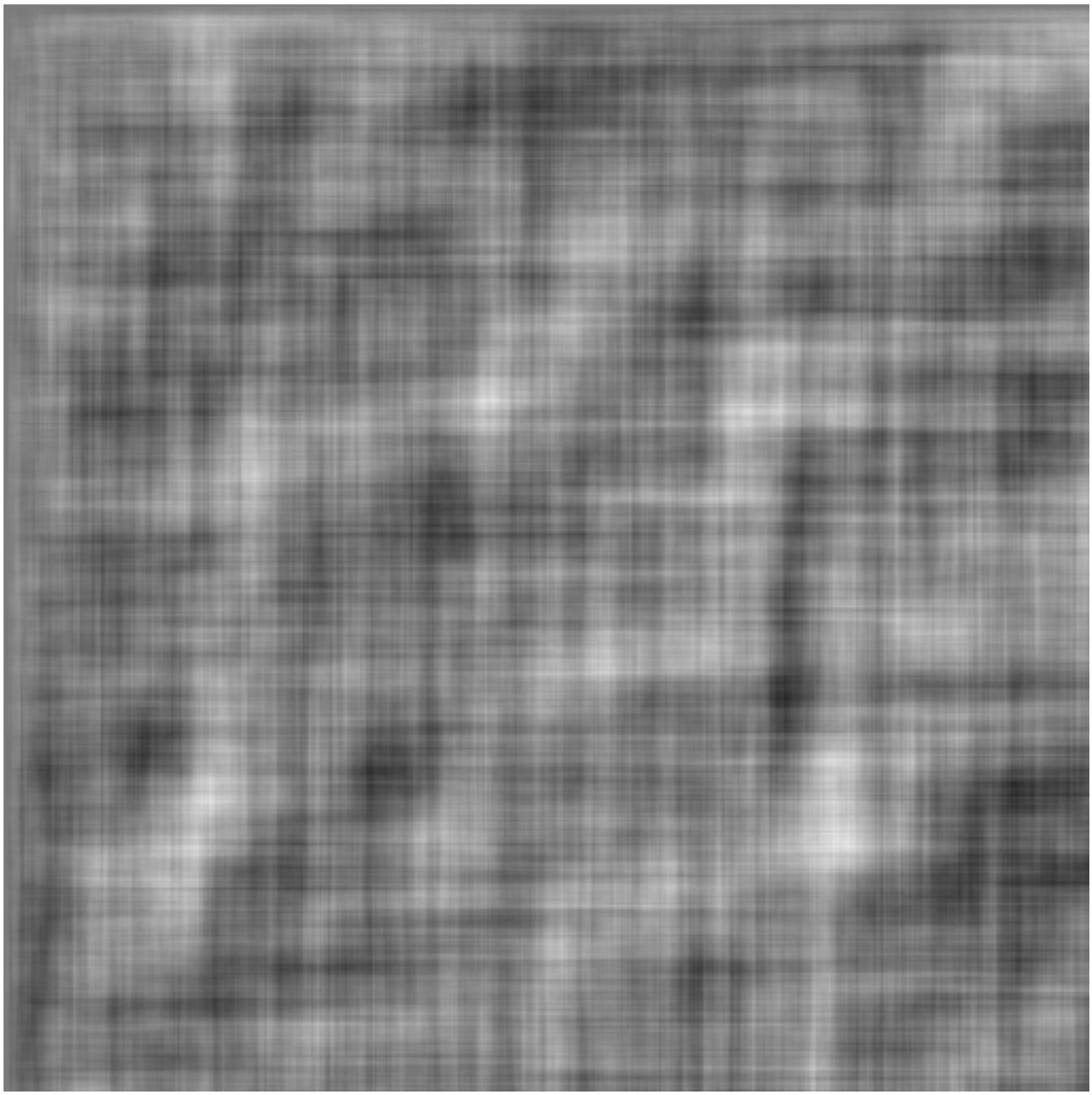


Figure 5: IIR Filtered image

3. Mesh plot of log $S_y(e^{j\mu}, e^{j\nu})$:

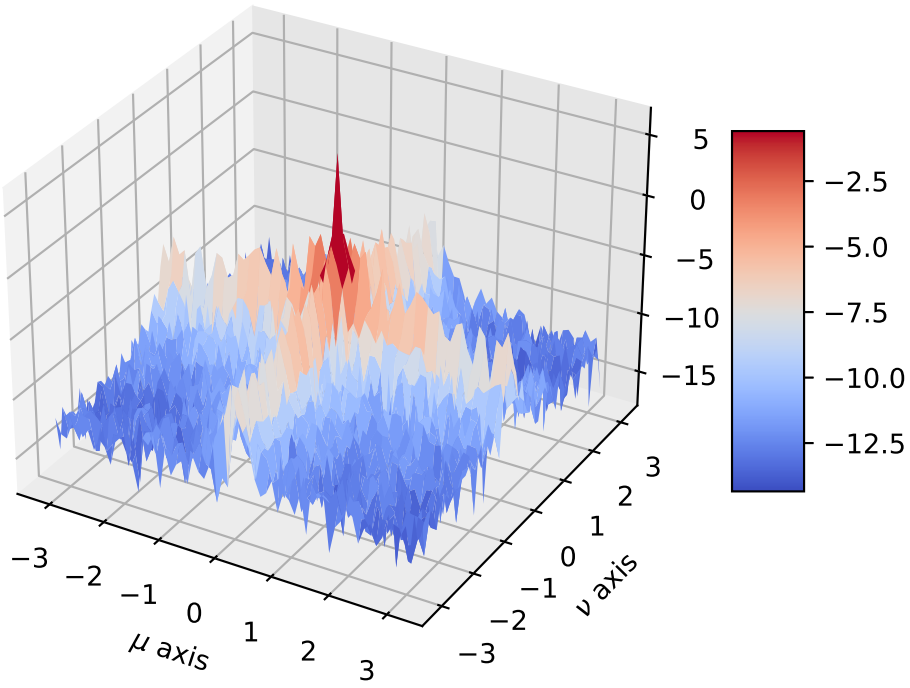


Figure 6: Mesh plot of PSD of y

4. Mesh plot of log of estimated PSD of y using *BetterSpecAnal(y)*:

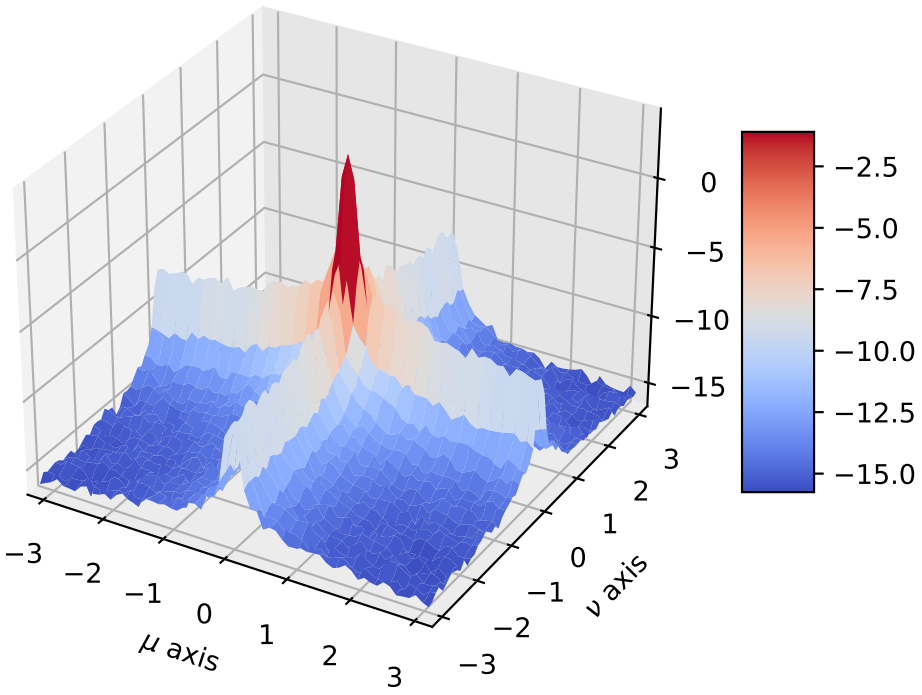


Figure 7: Mesh plot of PSD of y using *BetterSpecAnal(x)*

Source Code

Listing 1: Python function for BetterSpecAnal

```
def BetterSpecAnal(x,base_name):
    block_size=64
    h,w = x.shape
    cx=w//2; cy=h//2
    or_x = cx-5*(block_size//2)
    or_y = cy-5*(block_size//2)
    windows = []
    for i in range(5):
        for j in range(5):
            ul_x = or_x +i*block_size
            ul_y = or_y +j*block_size
            windows.append(x[ul_y:ul_y+block_size,ul_x:ul_x+block_size])
    W = np.hamming(block_size)
    W = np.outer(W,W)
    #multiply 2D hamming window
    windows = [w*W for w in windows]
    #compute squred DFT magnitude
    Z = [ (1/(block_size**2)*np.abs(np.fft.fft2(w))**2) for w in windows]
    Z = [np.fft.fftshift(z) for z in Z]
    Zabs = [np.log(z) for z in Z]
    #compute average
    av = np.zeros((block_size,block_size))
    for z in Zabs: av+=z
    av/=25

    # Plot the result using a 3-D mesh plot and label the x and y axises properly.
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    a = b = np.linspace(-np.pi, np.pi, num = block_size)
    X, Y = np.meshgrid(a, b)

    surf = ax.plot_surface(X, Y, av, cmap=plt.cm.coolwarm)
    ax.set_xlim(-1*np.pi,1*np.pi)
    ax.set_ylim(-1*np.pi,1*np.pi)
    ax.autoscale(enable=True,axis='z',tight=True)
    ax.set_xlabel('$\mu$ axis')
    ax.set_ylabel('$\\nu$ axis')
    ax.set_zlabel('Z Label')

    fig.colorbar(surf, shrink=0.5, aspect=5)
    plt.savefig('BetterSpecAnal_'+base_name+'.eps',format='eps')
```

Appendix

Listing 2: Complete python code for computing PSD

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab2- 2D Random Process
"""
```

```
8
9  import numpy as np                  # Numpy is a library support computation of large
       , multi-dimensional arrays and matrices.
10 from PIL import Image                # Python Imaging Library (abbreviated as PIL) is
       a free and open-source additional library for the Python programming language
       that adds support for opening, manipulating, and saving many different image
       file formats.
11 import matplotlib.pyplot as plt     # Matplotlib is a plotting library for the Python
        programming language.
12 import sys,os
13
14
15 def SpecAnal(x,cx,cy,block_size,base_name):
16     i = cx
17     j = cy
18     N = block_size
19
20     z = x[i:N+i, j:N+j]
21
22     # Compute the power spectrum for the NxN region.
23     Z = (1/N**2)*np.abs(np.fft.fft2(z))**2
24
25     # Use fftshift to move the zero frequencies to the center of the plot.
26     Z = np.fft.fftshift(Z)
27
28     # Compute the logarithm of the Power Spectrum.
29     Zabs = np.log(Z)
30
31     # Plot the result using a 3-D mesh plot and label the x and y axises properly.
32     fig = plt.figure()
33     ax = fig.add_subplot(111, projection='3d')
34     a = b = np.linspace(-np.pi, np.pi, num = N)
35     X, Y = np.meshgrid(a, b)
36
37     surf = ax.plot_surface(X, Y, Zabs, cmap=plt.cm.coolwarm)
38
39     ax.set_xlabel('$\mu$ axis')
40     ax.set_ylabel('$\\nu$ axis')
41     ax.set_zlabel('Z Label')
42
43     fig.colorbar(surf, shrink=0.5, aspect=5)
44
45     plt.savefig('SpecAnal_'+base_name+'_'+str(block_size)+'.eps',format='eps')
46
47 def BetterSpecAnal(x,base_name):
48     block_size=64
49     h,w = x.shape
50     cx=w//2; cy=h//2
51     or_x = cx-5*(block_size//2)
52     or_y = cy-5*(block_size//2)
53     windows = []
54     for i in range(5):
55         for j in range(5):
56             ul_x = or_x +i*block_size
57             ul_y = or_y +j*block_size
58             windows.append(x[ul_y:ul_y+block_size,ul_x:ul_x+block_size])
59     W = np.hamming(block_size)
60     W = np.outer(W,W)
61     #multiply 2D hamming window
```

```
62      windows = [w*W for w in windows]
63      #compute squred DFT magnitude
64      Z = [ (1/(block_size**2)*np.abs(np.fft.fft2(w))**2) for w in windows]
65      Z = [np.fft.fftshift(z) for z in Z]
66      Zabs = [np.log(z) for z in Z]
67      #compute average
68      av = np.zeros((block_size,block_size))
69      for z in Zabs: av+=z
70      av/=25
71
72      # Plot the result using a 3-D mesh plot and label the x and y axises properly.
73      fig = plt.figure()
74      ax = fig.add_subplot(111, projection='3d')
75      a = b = np.linspace(-np.pi, np.pi, num = block_size)
76      X, Y = np.meshgrid(a, b)
77
78      surf = ax.plot_surface(X, Y, av, cmap=plt.cm.coolwarm)
79      ax.set_xlim(-1*np.pi,1*np.pi)
80      ax.set_ylim(-1*np.pi,1*np.pi)
81      ax.autoscale(enable=True,axis='z',tight=True)
82      ax.set_xlabel('$\mu$ axis')
83      ax.set_ylabel('$\\nu$ axis')
84      ax.set_zlabel('Z Label')
85
86      fig.colorbar(surf, shrink=0.5, aspect=5)
87      plt.savefig('BetterSpecAnal_'+base_name+'.eps',format='eps')
88
89
90
91  if __name__=="__main__":
92      # Read in a gray scale TIFF image.
93      img_name=sys.argv[1]
94      base_name=os.path.basename(img_name).split('.')[0]
95      flag_betterspecanal = int(sys.argv[2])
96      if(flag_betterspecanal==0):
97          print('Enter BlockSize: ')
98          block_size=int(input())
99
100     im = Image.open(img_name)
101     print('Read '+img_name)
102     print('Image size: ', im.size)
103
104     # Display image object by PIL.
105     im.show(title='image')
106
107     # Import Image Data into Numpy array.
108     # The matrix x contains a 2-D array of 8-bit gray scale values.
109     x = np.array(im)
110     print('Data type: ', x.dtype)
111
112     # Display numpy array by matplotlib.
113     plt.imshow(x, cmap=plt.cm.gray)
114     plt.title('Image')
115
116     # Set colorbar location. [left, bottom, width, height].
117     cax =plt.axes([0.9, 0.15, 0.04, 0.7])
118     plt.colorbar(cax=cax)
119     plt.show()
120
```

```
    x = np.double(x)/255.0
    if flag_betterspecanal==0:
        SpecAnal(x,99,99,block_size,base_name)
    else:
        BetterSpecAnal(x,base_name)
```

Listing 3: Python code for generating filtered image for section 2

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab2 2-D Random Process
"""

from PIL import Image as pil
import numpy as np
import sys
import matplotlib.pyplot as plt

A=3.0
B=0.99
C=-0.981

def IIR_filter(x):
    y=np.zeros_like(x)
    h,w = x.shape
    for m in range(1,h):
        for n in range(1,w):
            y[m][n] = A*x[m][n] +B*(y[m-1][n]+y[m][n-1]) +C*y[m-1][n-1]
    return y

def main(fname):
    #create random image
    x = np.random.rand(512,512)
    x -= 0.5 # now in [-0.5,0.5]
    #display scaled image
    x_scaled = 255*(x+0.5)
    plt.imshow(x_scaled.astype(np.uint8))
    img_out = pil.fromarray(x_scaled.astype(np.uint8))
    img_out.save('Random_image.tif')
    #filter image
    y = IIR_filter(x)
    plt.imshow((y+127).astype(np.uint8))
    img_out = pil.fromarray((y+127).astype(np.uint8))
    img_out.save(fname)

if __name__=="__main__":
    fname = sys.argv[1]
    main(fname)
```