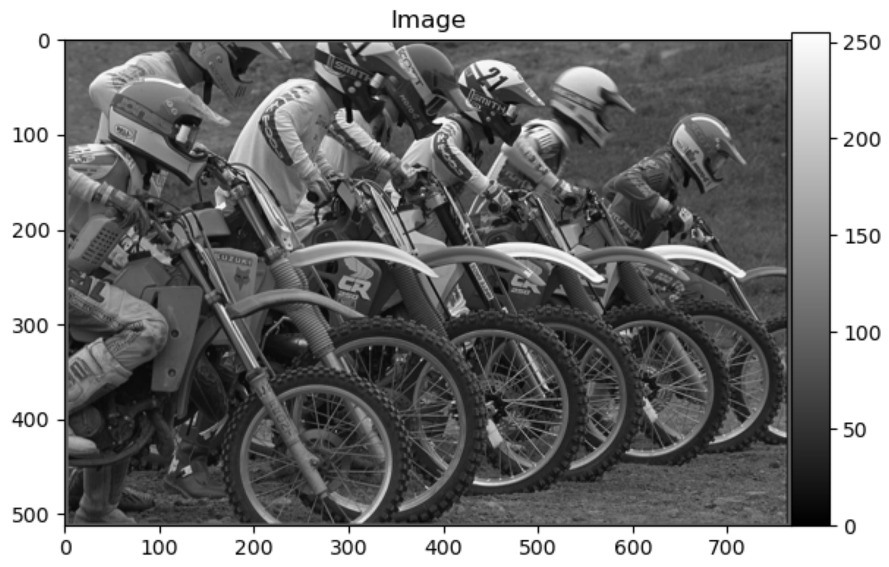
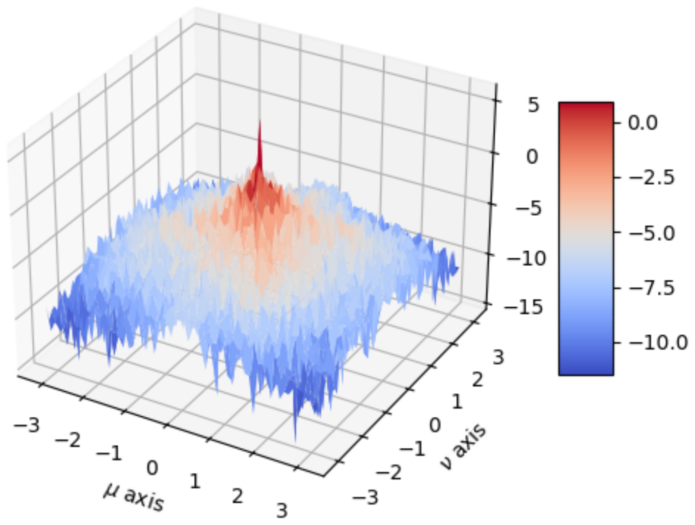
Report Lab 2 – Misha Tsysin, 0033922418

Power Spectral Density of an Image

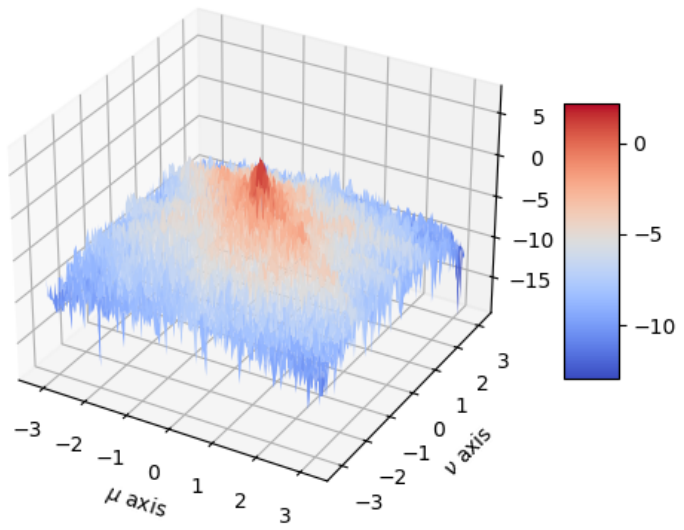
The grayscale image img04g.tif looks like:



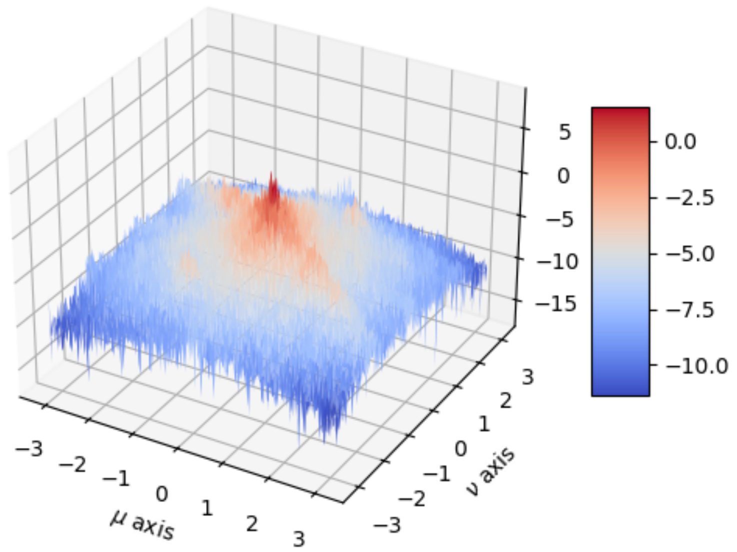
The PSD plots are:
For block size 64:



128:

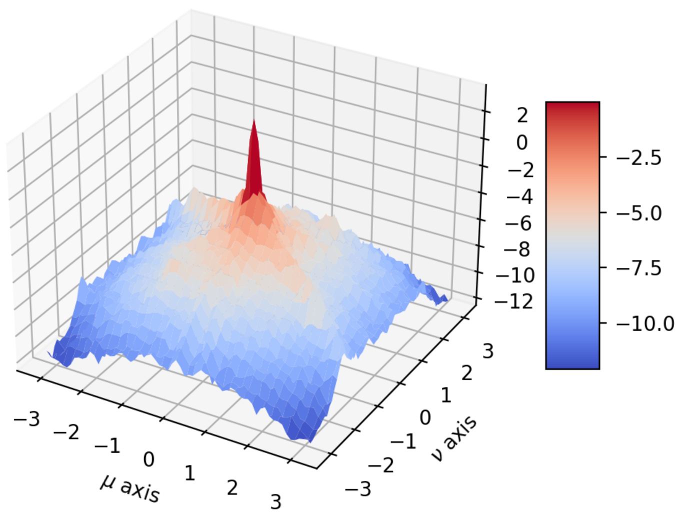


256:



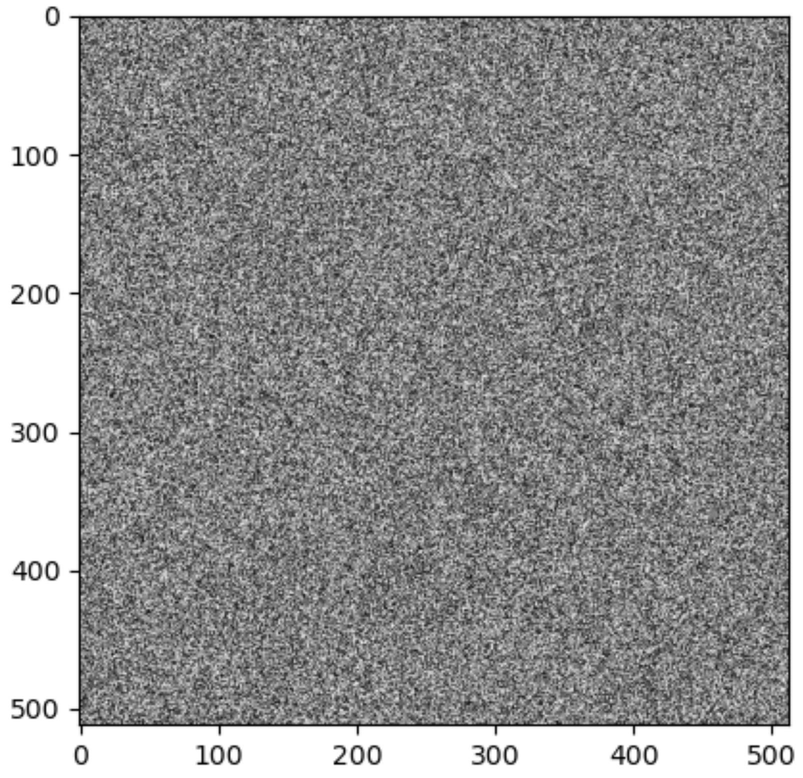
We see that the PSD is still noisy no matter the size of the block.

The improved PSD estimate is a lot less noisy:



Power Spectral Density of a 2-D AR Process

The 2d Gaussian random noise will look like:



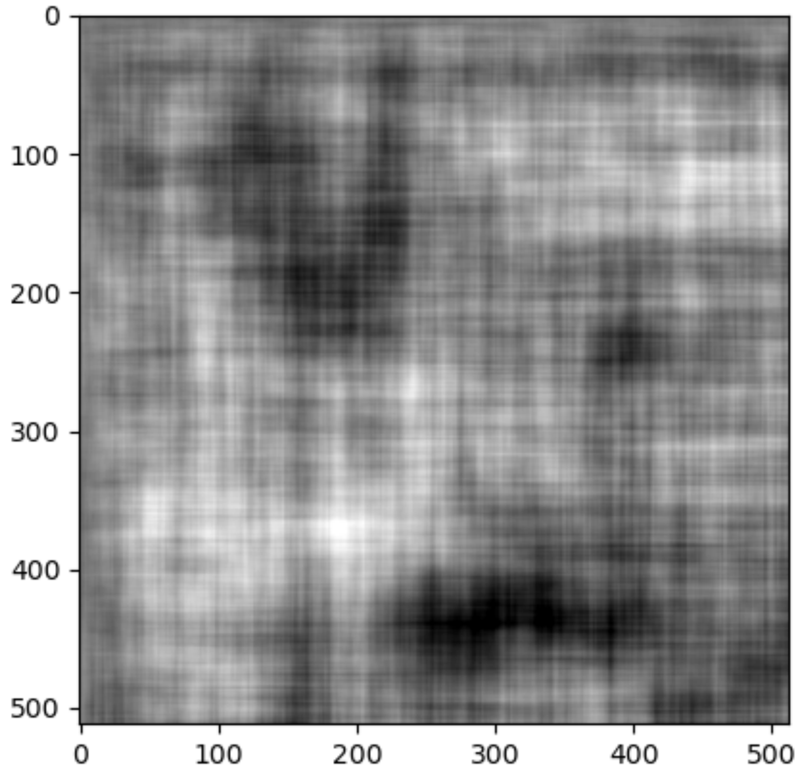
Now we can derive the difference equation for the given filter:

$$H(z_1, z_2) = \frac{Y(z_1, z_2)}{X(z_1, z_2)} = \frac{3}{1 - 0.99(z_1^{-1} + z_2^{-1}) + 0.9801z_1^{-1}z_2^{-1}}$$

By multiplying out and taking the inverse Z transform we can get:

$$y(m, n) = 3x(m, n) + 0.99\big(y(m, n-1) + y(m-1, n)\big) - 0.9801y(m-1, n-1)$$

ss
Applying this to the input noise, we can get:



The theoretical PSD for this transfer function is equal to:

$$S_y\left(e^{j\mu}, e^{j\nu}\right) = \left|H\left(e^{j\mu}, e^{j\nu}\right)\right|^2 S_y\left(e^{j\mu}, e^{j\nu}\right)$$

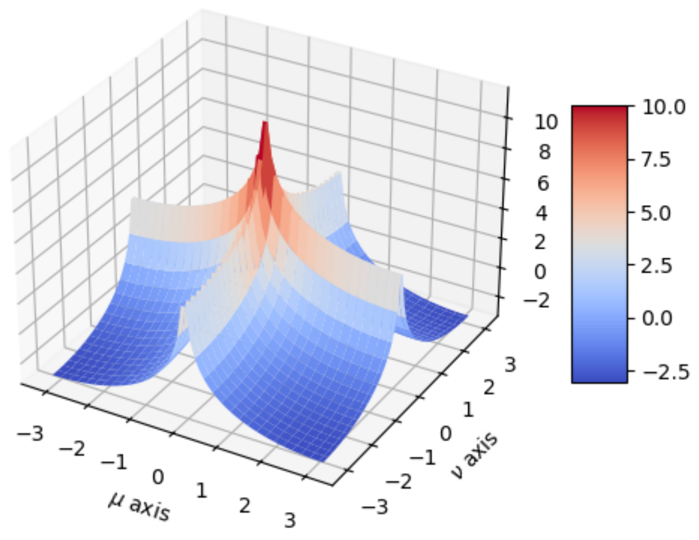
We know the transfer function. Thus, we just need to find an expression for the original PSD S_x. We know that the PSD of a white noise is just a constant. Thus we can simply look at the variance of each of the gaussians we use for each pixel. For a uniform random variable distributed in [-0.5, 0.5] the variance is:

$$Var(x) = E[x^2] - E[x]^2 = E[x^2] = \int_{-0.5}^{0.5} x^2 dx = 2 * \frac{0.5^3}{3} = \frac{1}{12}$$

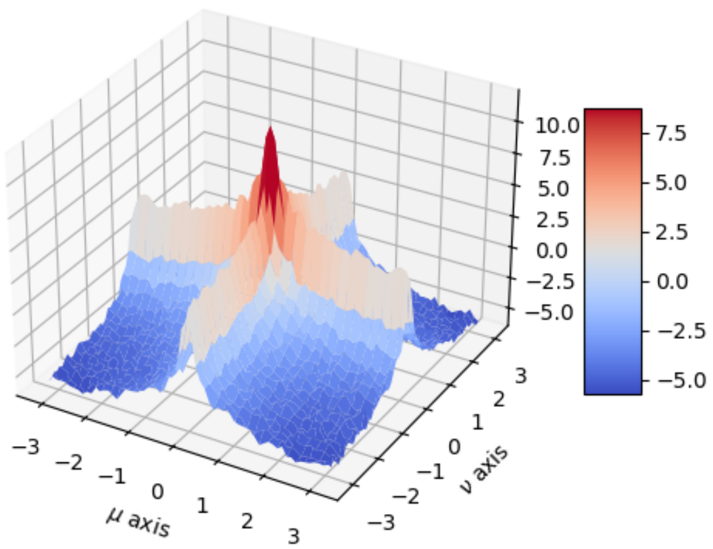
Thus the final answer is:

$$S_y\left(e^{j\mu}, e^{j\nu}\right) = \frac{1}{12}\left|\frac{3}{1 - 0.99(e^{-i\mu} + e^{-i\nu}) + 0.9801e^{-i\mu}e^{-i\nu}}\right|^2$$

The theoretical PSD will look like:



While the one obtained by BetterSpecAnal is:



Code:

BetterSpecAnal.py:

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def BetterSpecAnal(x: np.ndarray, N: int, n: int):
    W = np.outer(np.hamming(N), np.hamming(N))

    h, w = x.shape

    h_begin = int((h - n * N) / 2)
    w_begin = int((w - n * N) / 2)

    res = np.zeros((N, N))
    for i in range(n):
        for j in range(n):
            t = W * x[h_begin + i * N: h_begin + (i+1) * N,
                      w_begin + j * N: w_begin + (j+1) * N]
            res += (1 / N**2) * np.abs(np.fft.fftshift(np.fft.fft2(t) ** 2)) * (1/n**2)

    return np.log(res)

if __name__ == "__main__":


    im = Image.open('img04g.tif')
    print('Read img04.tif.')
    print('Image size: ', im.size)

    # Import Image Data into Numpy array.
    # The matrix x contains a 2-D array of 8-bit gray scale values.
    x = np.double(np.array(im))/255.0
    N, n = 64, 5

    # 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 = N)
    X, Y = np.meshgrid(a, b)

    PSD = BetterSpecAnal(x, N, n)

    surf = ax.plot_surface(X, Y, PSD, cmap=plt.cm.coolwarm)
    ax.set_xlabel('$\mu$ axis')
    ax.set_ylabel('$\\nu$ axis')
    ax.set_zlabel('Z Label')

    fig.colorbar(surf, shrink=0.5, aspect=5)

    plt.show()
```

Part2.py

```
import numpy as np
import matplotlib.pyplot as plt
from BetterSpecAnal import BetterSpecAnal

N = 512
x = np.random.uniform(-0.5, 0.5, (N, N))
x_scaled = 255 * (x + 0.5)

plt.imshow(np.uint8(x_scaled), cmap=plt.cm.gray)
plt.show()

y = np.zeros((N, N))

for i in range(N):
    for j in range(N):
        y[i, j] += 3 * x[i, j]
        if i > 0:
            y[i, j] += 0.99 * y[i-1, j]
        if j > 0:
            y[i, j] += 0.99 * y[i, j-1]
        if i > 0 and j > 0:
            y[i, j] -= 0.9801 * y[i-1, j-1]
print(np.max(y))

plt.imshow(np.uint8(np.clip(y + 127, 0, 255)), cmap=plt.cm.gray)
plt.show()


s = BetterSpecAnal(y, 64, 5)
U, V = np.meshgrid(np.linspace(-np.pi, np.pi, 64), np.linspace(-np.pi, np.pi, 64))
print (U, V)
theoretical_s = (1/12) *  np.abs(3 / ((1 - 0.99 * np.exp(-1j * U)) * (1 - 0.99 * np.exp(-1j *
V)))) ** 2

# 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')
```

```python
a = b = np.linspace(-np.pi, np.pi, num = 64)
X, Y = np.meshgrid(a, b)

surf = ax.plot_surface(X, Y, s, cmap=plt.cm.coolwarm)

ax.set_xlabel('$\mu$ axis')
ax.set_ylabel('$\\nu$ axis')
ax.set_zlabel('Z Label')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()


# 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 = 64)
X, Y = np.meshgrid(a, b)

surf = ax.plot_surface(X, Y, np.log(theoretical_s), cmap=plt.cm.coolwarm)

ax.set_xlabel('$\mu$ axis')
ax.set_ylabel('$\\nu$ axis')
ax.set_zlabel('Z Label')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```