Advanced Digital Signal Processing (ADSP) Lab - Python Lab Manual

Course Code: EEE G613 **Instructer in Charge:** Dr. Rajesh Kumar Tripathy **Teaching Assistant:** Shaswati Dash **Lab Technician:** Ramesh Pokanati

---

Experiment No. - 6

Power Spectrum Estimation

Consider N=64 samples for an AR(4) process x(n) that is generated by filtering a unit variance Gaussian Noise v(n) with the filter transfer function given below

$$H(z) = \frac{b(0)}{1+\sum_{k=1}^{4} a(k) z^{-k}}$$

where b(0) = 1, a(1)= 0.7348, a(2) =1.8820, a(3) = 0.7057, a(4) = 0.8851

1 a) Obtain the autocorrelation for the signal x(n) which is the estimate

$$\hat{r}_x(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) x(n-k)$$

b) Generate the power spectrum for the above signal. Power spectrum equation given below

$$\hat{P}_x(e^{jw}) = \sum_{-N}^{N} \hat{r}_x(k) e^{-jkw}$$

2) a) Using the estimated autocorrelation of x(n) by adapting Yule-Walker equations, obtain the filter coefficients b(0), a(1), a(2), a(3) and a(4)

b) Calculate the power spectrum using the expression given below which is based on the filter coefficients

$$\hat{P}_x(e^{jw}) = \frac{|b(0)|^2}{\left|1+\sum_{k=1}^{p} a(k) e^{-jkw}\right|^2}$$

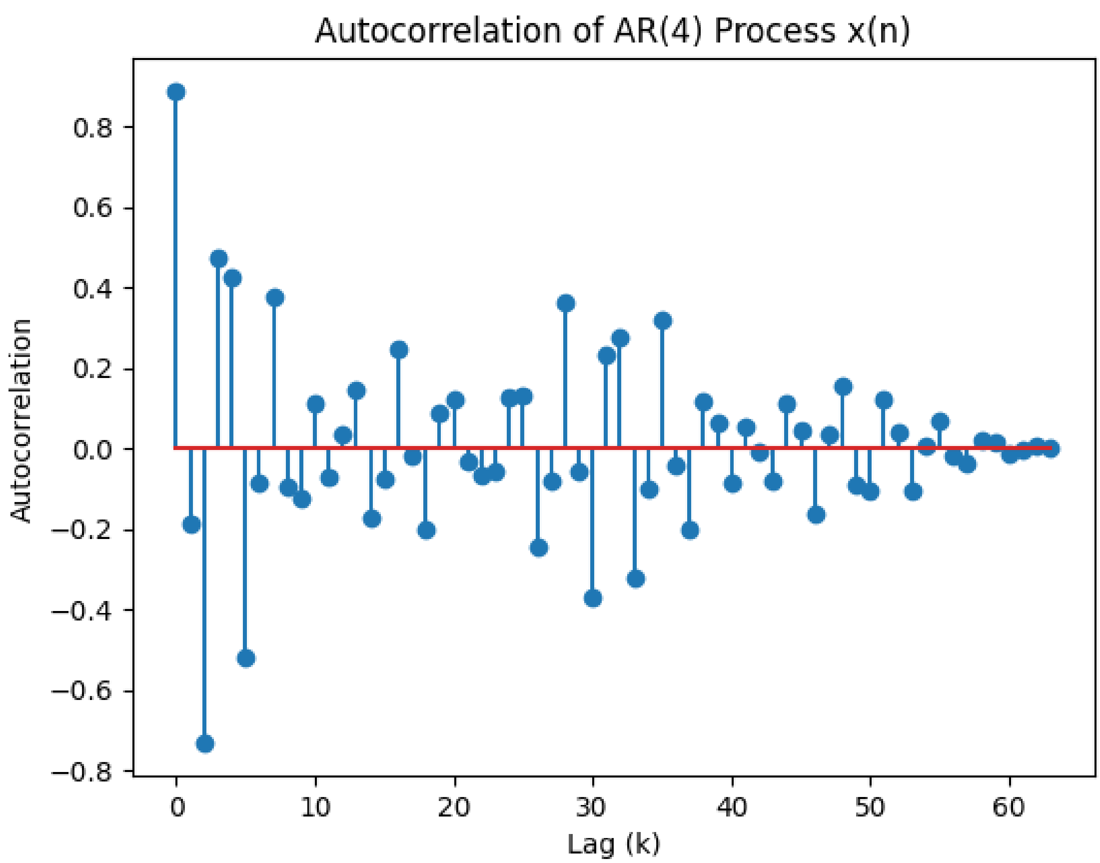
Compare the power spectrum obtained in 1 b) with 2 b)

Python Code-

```
#import libraries
import numpy as np
```

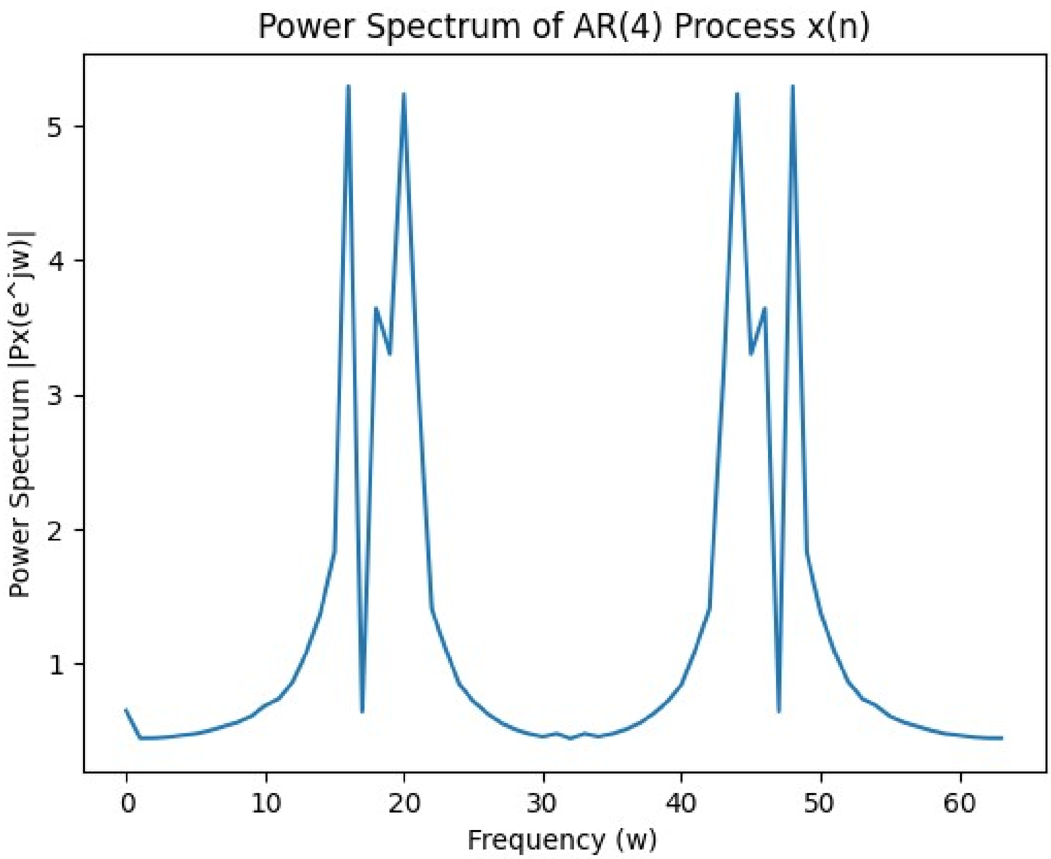
```
import matplotlib.pyplot as plt
from scipy.signal import lfilter

#part-1(a)
#obtain the autocorrelation for the signal x(n)
np.random.seed(0) # Set a random seed for reproducibility
a = [1, 0.7348, 1.8820, 0.7057, 0.8851]
b = [1]
N = 64
v = np.random.rand(N)
x = lfilter(b, a, v) # Filter the random signal
acv = np.correlate(x, x, mode="full") # Calculate the autocorrelation
acv = acv[N - 1:]
acv = acv/100
plt.stem(acv)
plt.xlabel('Lag (k)')
plt.ylabel('Autocorrelation')
plt.title('Autocorrelation of AR(4) Process x(n)')
plt.show()
```




```
#part-1(b)
#generate the power spectrum for the above signal
```

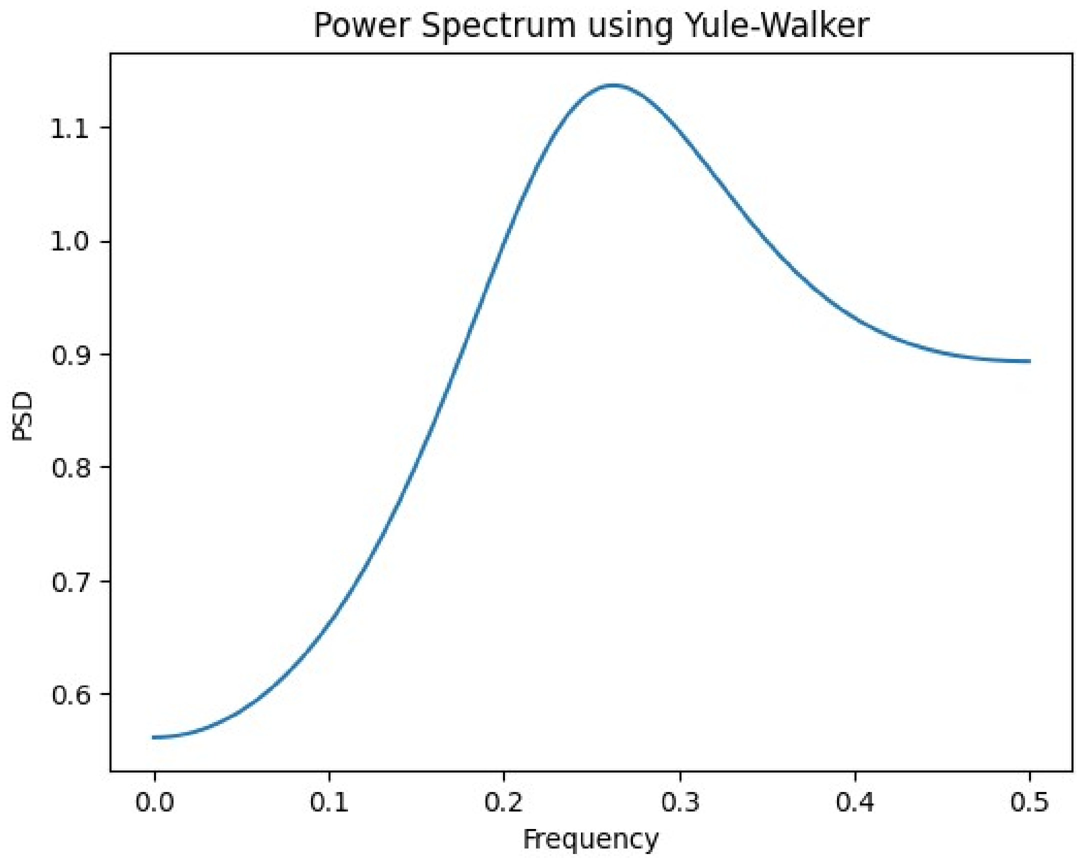
```
psd = np.fft.fft(acv)
plt.figure()
plt.plot(np.abs(psd))
plt.xlabel('Frequency (w)')
plt.ylabel('Power Spectrum |Px(e^jw)|')
plt.title('Power Spectrum of AR(4) Process x(n)')
plt.show()
```




```
#part-2
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz
np.random.seed(0) # Set a random seed for reproducibility
a = [1, 0.7348, 1.8820, 0.7057, 0.8851]
b = [1]
N = 64
v = np.random.rand(N)
x = lfilter(b, a, v) # Filter the random signal
acv = np.correlate(x, x, mode='full') # Calculate the autocorrelation
acv = acv[N - 1:]
M = 4
R = np.array([
```

```
[acv[0], acv[1], acv[2], acv[3]],
[acv[1], acv[0], acv[1], acv[2]],
[acv[2], acv[1], acv[0], acv[1]],
[acv[3], acv[2], acv[1], acv[0]]
])
bmat = np.array([-acv[1], -acv[2], -acv[3], -acv[4]])
coeff = np.linalg.solve(R, bmat)
print(f'b(0): {b[0]}')
for i in range(M):
  print(f'a({i + 1}): {coeff[i]}')
Nfft = 2048 # Number of points for the frequency response
W, H = freqz(b, coeff, worN=Nfft)
plt.figure()
plt.plot(W / (2 * np.pi), abs(H))
plt.xlabel("Frequency")
plt.ylabel("PSD")
plt.title("Power Spectrum using Yule-Walker")
plt.show()
```

```
b(0): 1
a(1): 0.30600956209319
a(2): 1.1447067214482551
a(3): 0.024655377929276485
a(4): 0.30593055518463347
```


Power Spectrum using Yule-Walker
PSD
Frequency
1.1
1.0
0.9
0.8
0.7
0.6
0.0
0.1
0.2
0.3
0.4
0.5