**EECS240: Random Processes**

Computer Project #2

Name: Chen Yang

ID: 29803931

**Problem1:**

**Consider an** **autoregressive (AR) random process Yn = αYn−1 + Xn where Xn is a White Gaussian noise with zero mean and variance σX2**

**(a) Plot some sample realizations of the above AR random process when σX2 = 1−α2, for example for α = 0.3 and α = 0.95.**

**<code>**

**Situation1:(α = 0.3)**

import numpy  
import matplotlib.pyplot as plt  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.3  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1  
  
plt.xlabel('Autoregressive Random Process')  
plt.ylabel('Value')  
plt.plot(Y)  
plt.show()

**Situation2:(α = 0.95)**

import numpy  
import matplotlib.pyplot as plt  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.95  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1  
  
plt.xlabel('Autoregressive Random Process')  
plt.ylabel('Value')  
plt.plot(Y)  
plt.show()

**<result>**

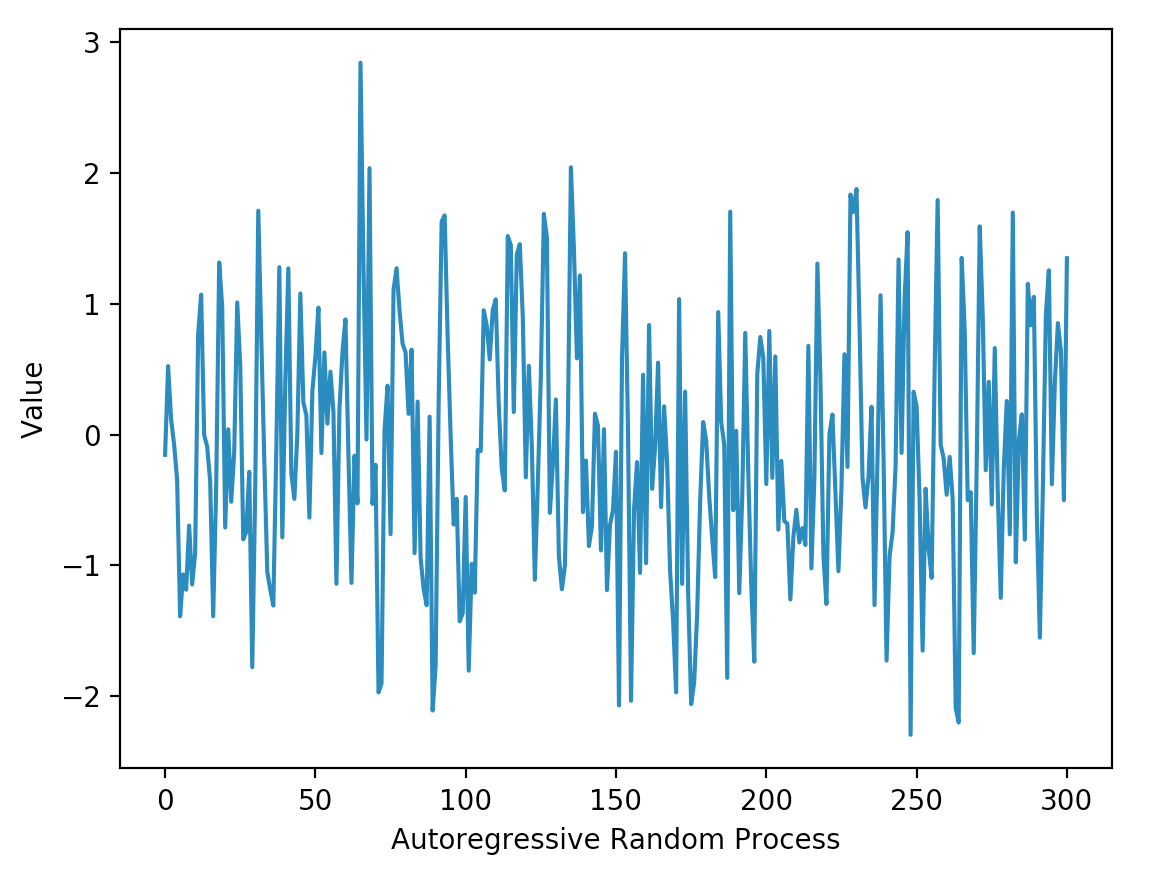


Figure1.1 autoregressive (AR) random process α = 0.3

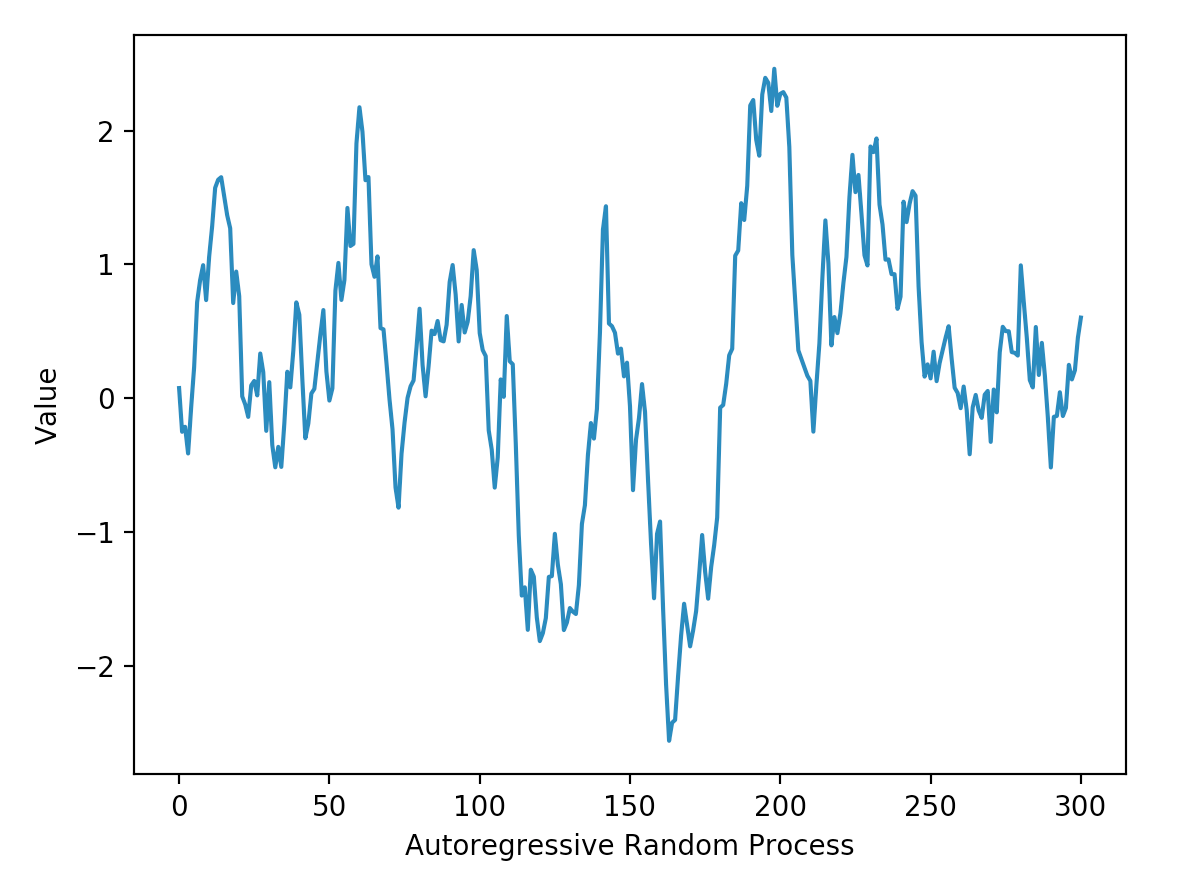


Figure1.2 autoregressive (AR) random process α = 0.95

**<Analysis>**

As you can see in the results, the two autoregressive random process with different parameters α = 0.3 and α = 0.95 have different figures. In the coding, we generate 300 times random numbers to form the AR random process. Obviously, the lager α displays a more stationary status and smaller one is sharper. So we could draw the conclusion that with α becoming smaller, the AR random process is getting more stationary.

**(b) Simulate and plot the autocorrelation RY (k) for α = 0.3 and α = 0.95.**

**<code>**

**Situation1:(α = 0.3)**

import numpy  
import matplotlib.pyplot as plt  
from statsmodels.graphics.tsaplots import plot\_acf  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.3  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1

plot\_acf(Y)  
plt.show()

**Situation2:(α = 0.95)**

import numpy  
import matplotlib.pyplot as plt  
from statsmodels.graphics.tsaplots import plot\_acf  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.95  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1  
  
plot\_acf(Y)  
plt.show()

**<result>**

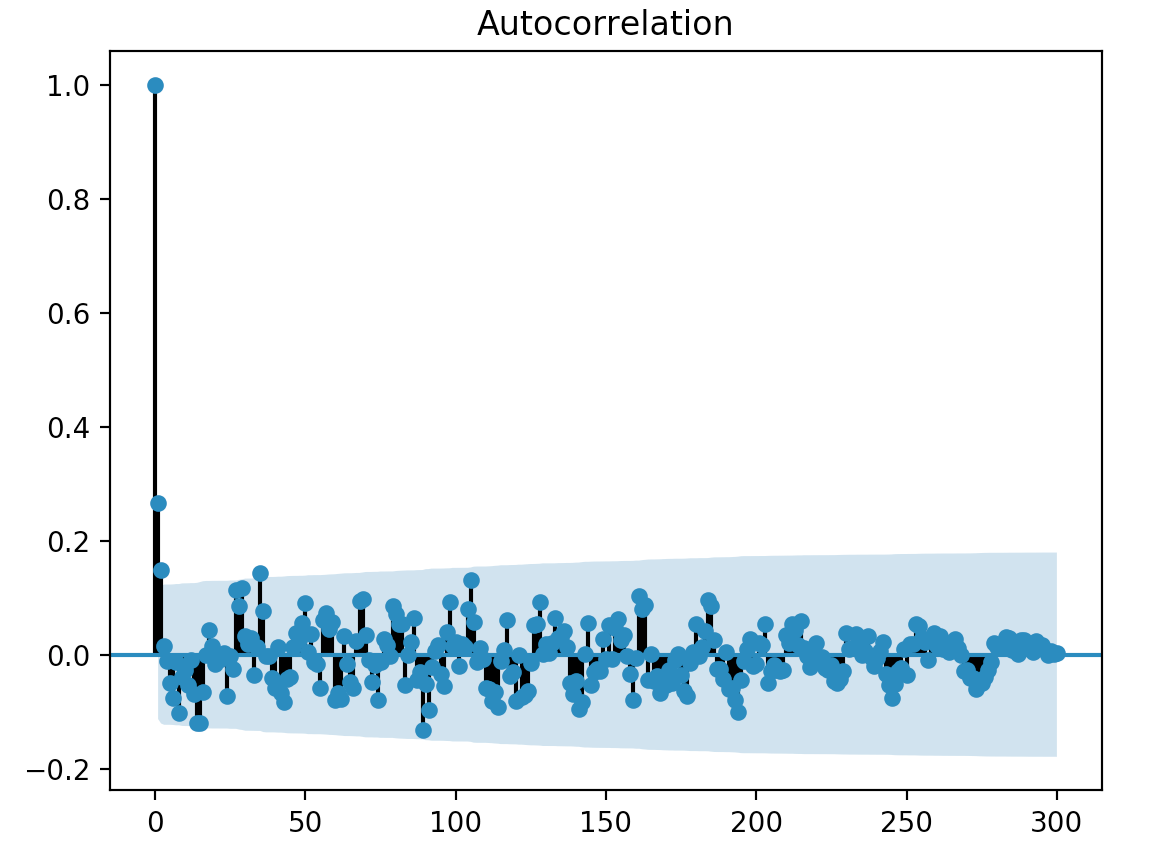
****

Figure1.3 autocorrelation RY (k) of AR random process α = 0.3

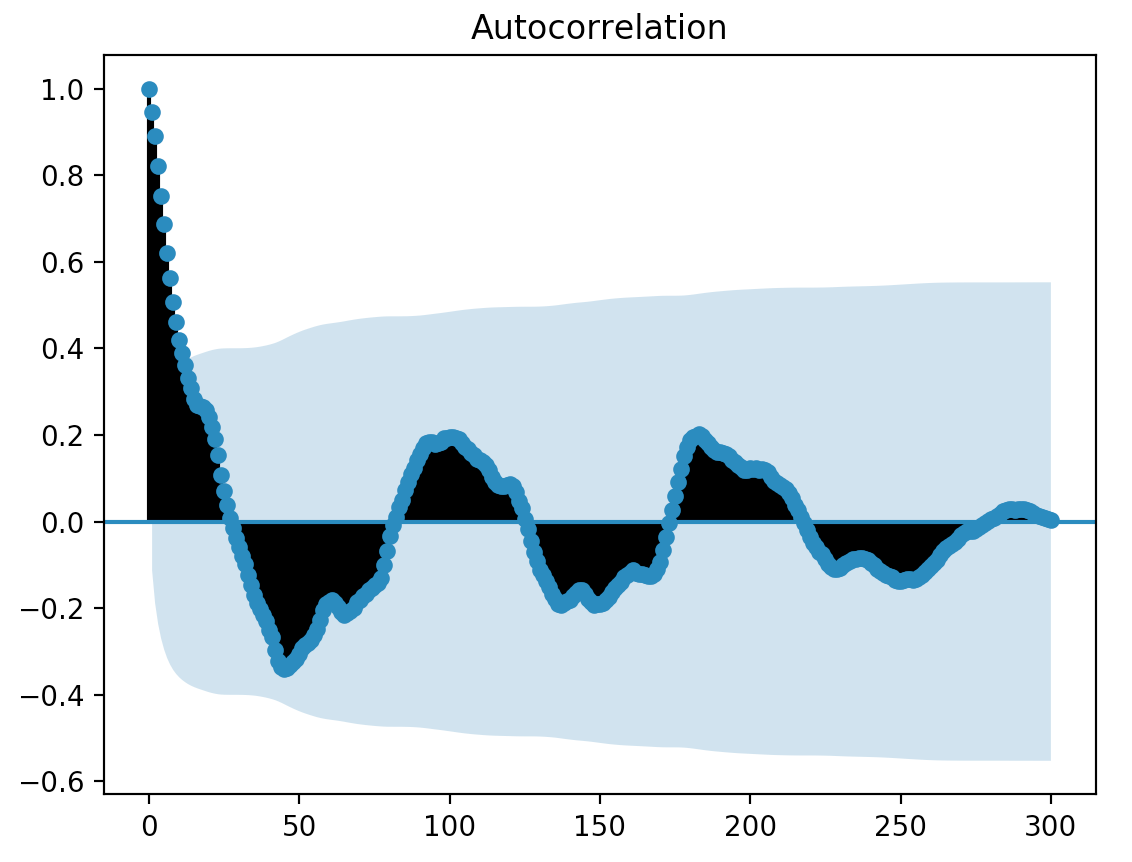


Figure1.4 autocorrelation RY (k) of AR random process α = 0.95

**<Analysis>**

In the results, the two diagram are totally different. The first one is Autocorrelation RY (k) with parameter α = 0.3, and the limits is from 0 to 300. With k becomes bigger, the RY (k) has a great change. But when α equals to 0.95, we could see that the change of RY (k) is getting more smooth, changing gradually.

**(c) Simulate and plot the power spectral density SY (f) for α = 0.3 and**

**α = 0.95**

**<code>**

**Situation1:(α = 0.3)**

import numpy  
import matplotlib.pyplot as plt  
from scipy import signal  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.3  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1  
sampling = 400  
freqs, P\_xx = signal.periodogram(Y, sampling, scaling = 'density')

plt.plot(freqs, P\_xx)  
plt.show()

**Situation2:(α = 0.95)**

import numpy  
import matplotlib.pyplot as plt  
from scipy import signal  
  
N = 300  
Y = numpy.zeros(N+1)  
mean = 0  
α = 0.95  
std = (1-α\*\*2)\*\*0.5  
Y[0] = numpy.random.normal(mean, std, size=1)  
i = 0  
while i <= N-1:  
 X = numpy.random.normal(mean, std, size=1)  
 Y[i+1] = α\*Y[i] + X  
 i = i + 1  
sampling = 400  
freqs, P\_xx = signal.periodogram(Y, sampling, scaling = 'density')

plt.plot(freqs, P\_xx)  
plt.show()

**<result>**

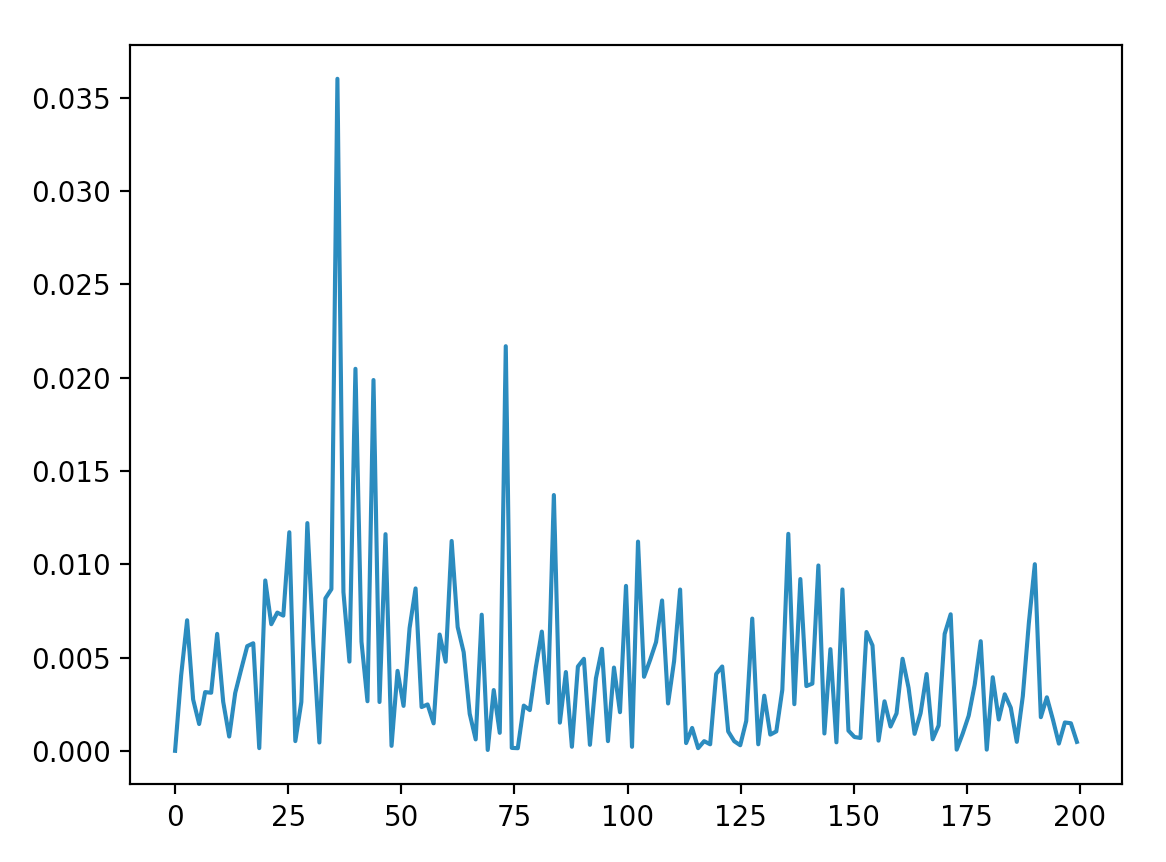


Figure1.5 power spectral density SY (f) for α = 0.3

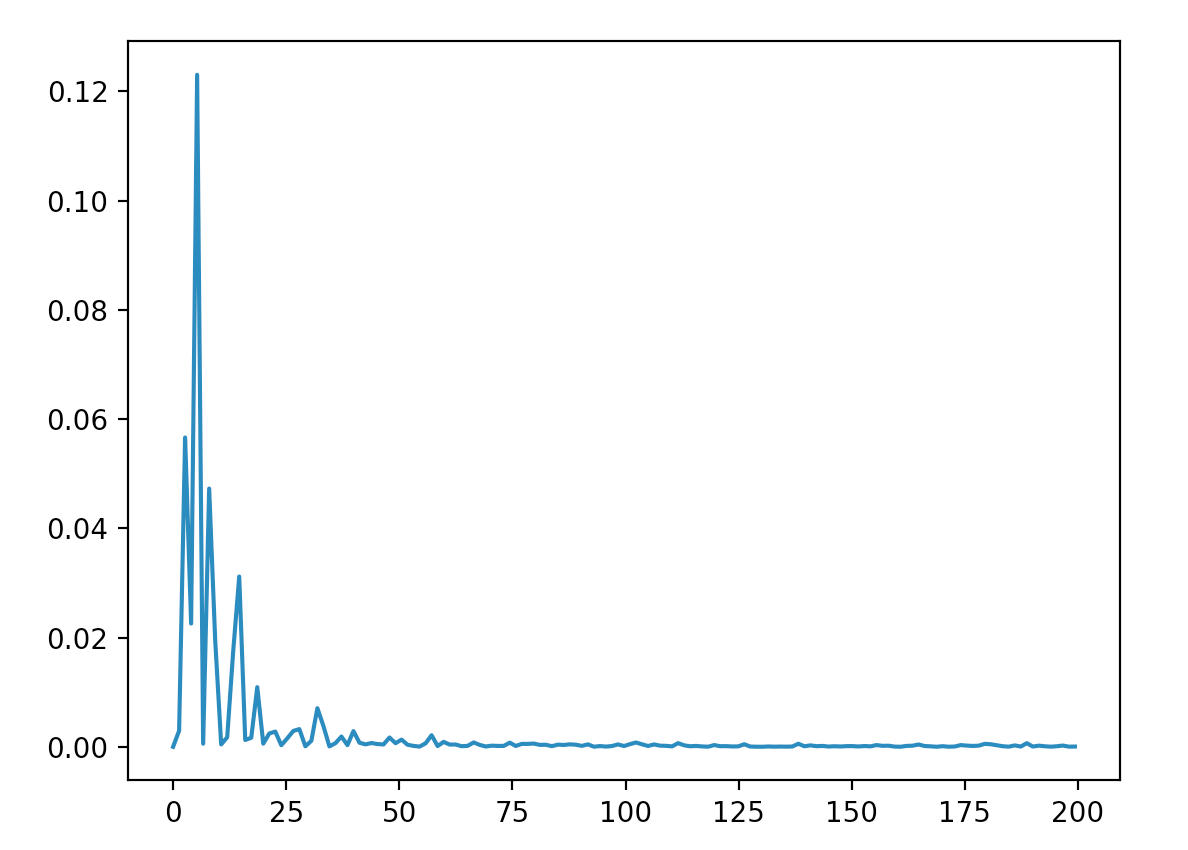


Figure1.6 power spectral density SY (f) for α = 0.95

**<Analysis>**

In the results, the lager parameter α = 0.95 makes the diagram more like a compressed one. And the smaller α = 0.3 makes it separately. So the power spectral density with smaller α looks like being distributed evenly and the lager α looks like only being concentrated in the frequency between 0 to 25Hz. Finally, we could make a conclusion that with the α becomes bigger, the power spectral density would be compressed into small scope.

**Problem2:**

**Consider the random process Xn = cos(0.2πn+Θ) where Θ is a uniform random variable between (−π, π). Draw one plot that contains 100 different realizations of Xn versus n (all in one plot). Use a dot to represent every (n,Xn) pair in the 2D plane.**

**<code>**

import numpy  
import matplotlib.pyplot as plt  
import math  
  
a = math.pi  
X = numpy.zeros(100)  
for i in range(99):  
 Θ = numpy.random.uniform(-a, a, 1)  
 X[i] = math.cos(0.2\*a\*i+Θ)  
  
plt.plot(X, 'o')  
plt.show()

**<result>**

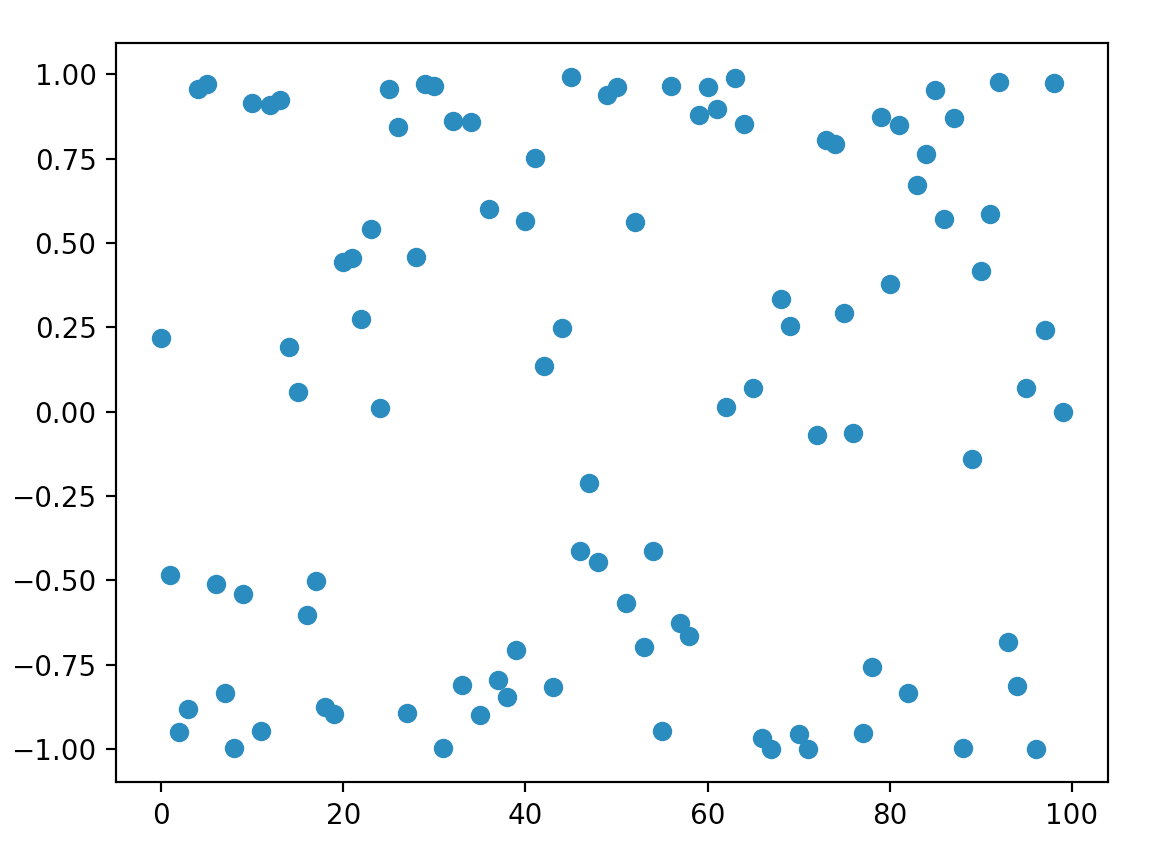


Figure2.1 (n,Xn) pairs in the 2D plane for random process Xn

**<Analysis>**

In this problem, we are required to generate a random process Xn = cos(0.2πn+Θ) where Θ is a uniform random variable between (−π, π). So firstly we generate a uniform random variable and then form the required random process. In every different time n we have a different RV and we plot it in the 2D plane. Shown in the above diagram.

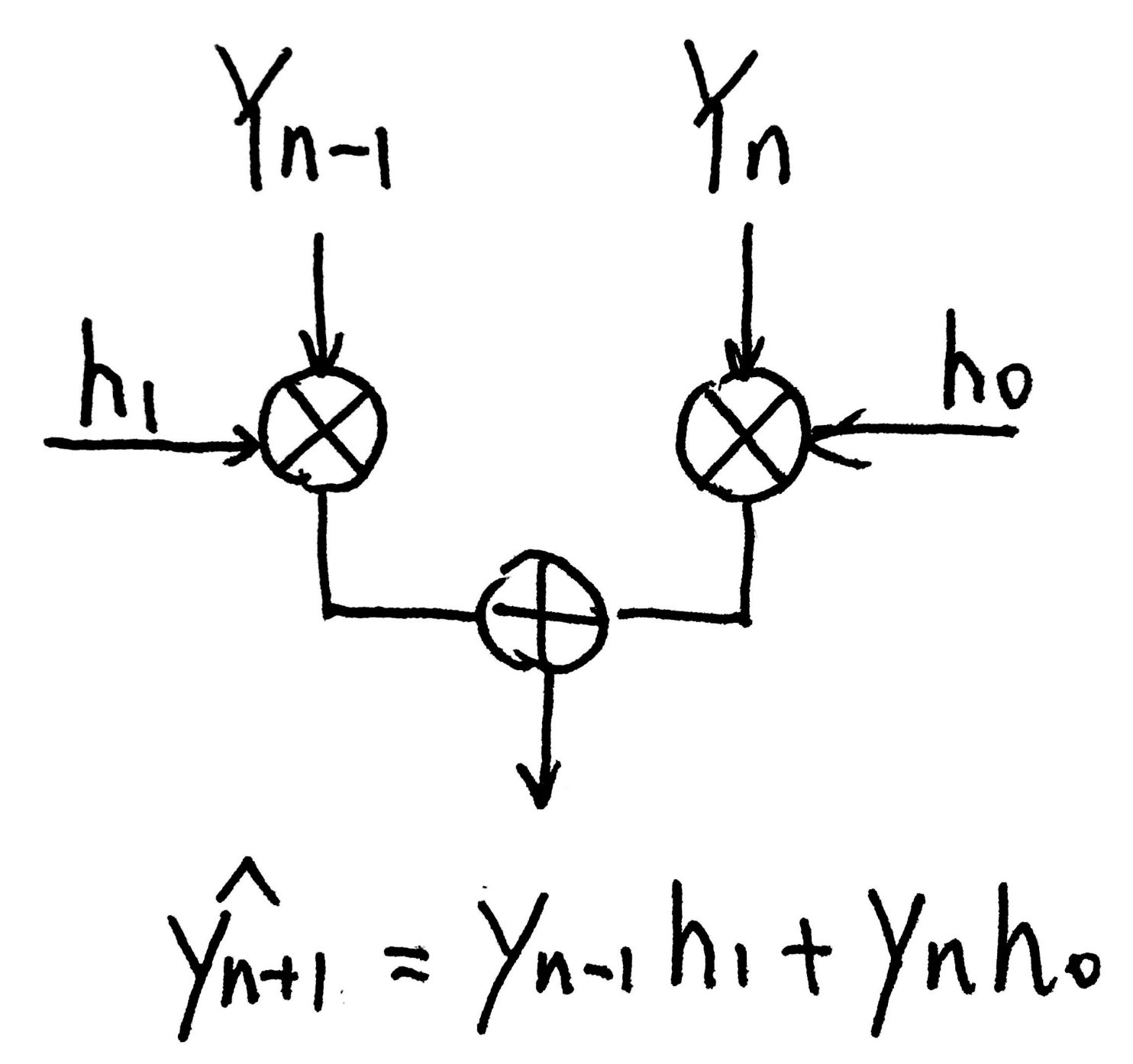
**Problem3:**

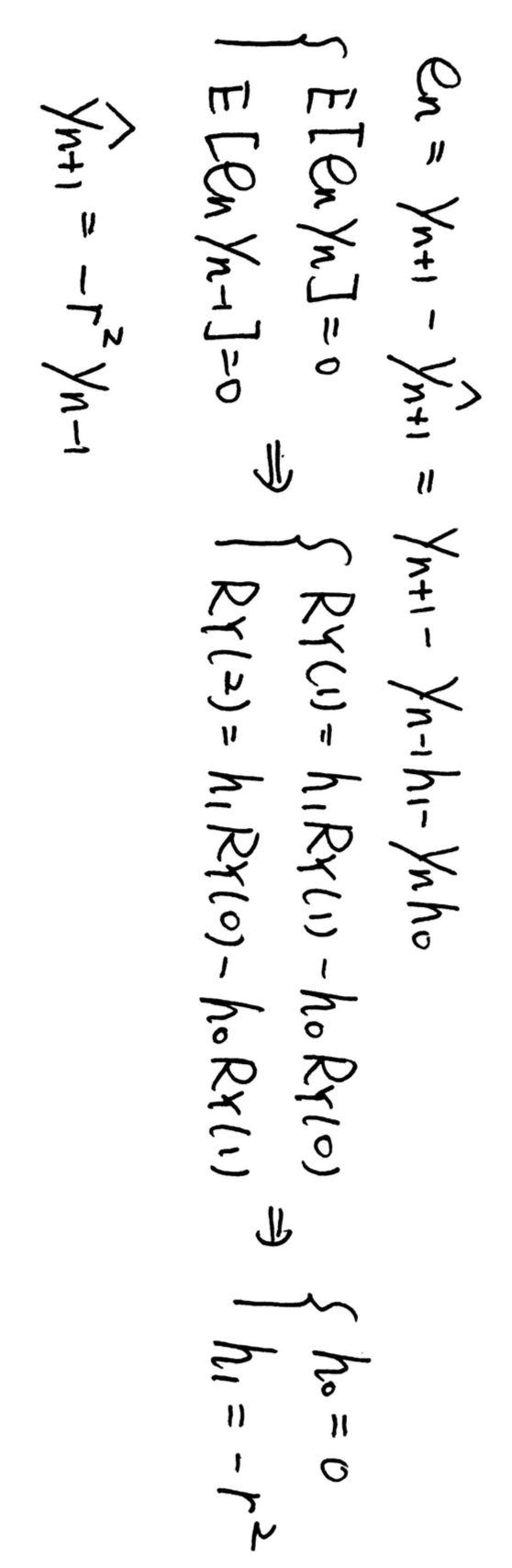
**Consider an AR(2) random process given by Yn = −r2Yn−2 + Xn where Xn is a White Gaussian noise with zero mean and variance σX2 and 0 < r < 1. The autocorrelation RY (k) = (σX2 /(1 − r4))r|k| cos(kπ/2).**

**(a) Find** **the** **optimal one-step linear predictor based on the present and past samples of Yn (****theoretically).**

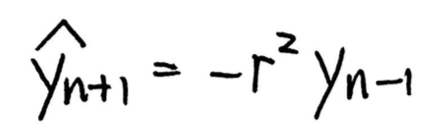
**<Theoretical Prove>**

As in the question above, it is said that the optimal one-step linear predictor is based on the present and past samples of Yn, so we should use Yn-1 and Yn to estimate Yn+1.

****

****

The equation:

****

is our optimal one-step linear predictor.

**(b) Consider σX2 = 1−r4 and two cases: (i) r = 0.5 and (ii) r = 0.95. Generate 150 samples of each process and discard the first 100 samples to make sure the generated samples are WSS. Then, plot the realization and its predicted values for each case. Which value of r results in a more predictable process?**

**<Code>**

import numpy  
import matplotlib.pyplot as plt  
import math  
  
N = 150  
Y = numpy.zeros(N+1)  
Z = numpy.zeros(N+1)  
mean = 0  
r = 0.95  
σX = math.sqrt(1-r\*r\*r\*r)  
Y[0] = numpy.random.normal(mean, σX, size=1)  
Y[1] = numpy.random.normal(mean, σX, size=1)  
Z[0] = numpy.random.normal(mean, σX, size=1)  
Z[1] = numpy.random.normal(mean, σX, size=1)  
for i in range(2, N):  
 X = numpy.random.normal(mean, σX, size=1)  
 Y[i] = -r\*r\*Y[i-2]+X  
for j in range(1, N-1):  
 Z[j+1] = -r\*r\*Y[j-1]  
  
plt.xlim(100, 150)  
plt.plot(Y, 'r')  
plt.plot(Z, 'b')  
plt.show()

**<Result>**

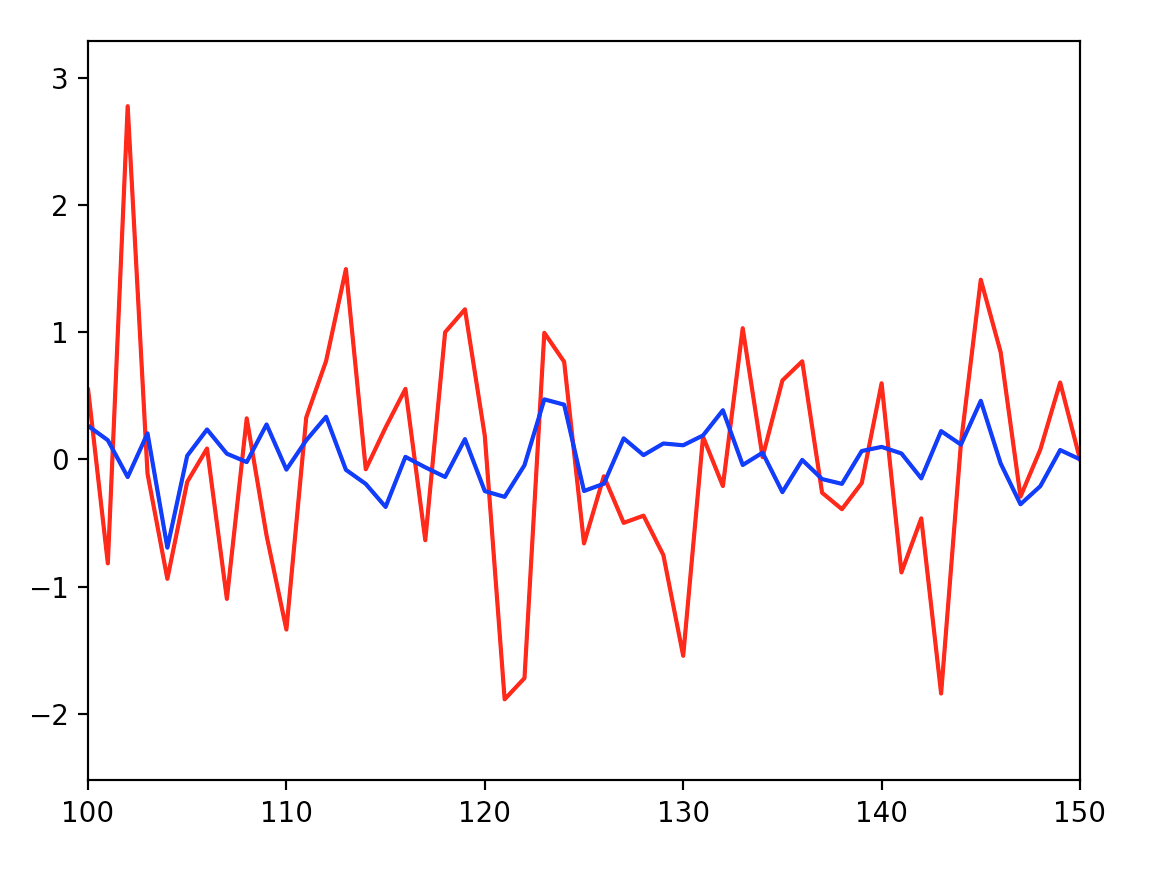
****

Figure3.1 optimal one-step linear predictor for r = 0.5

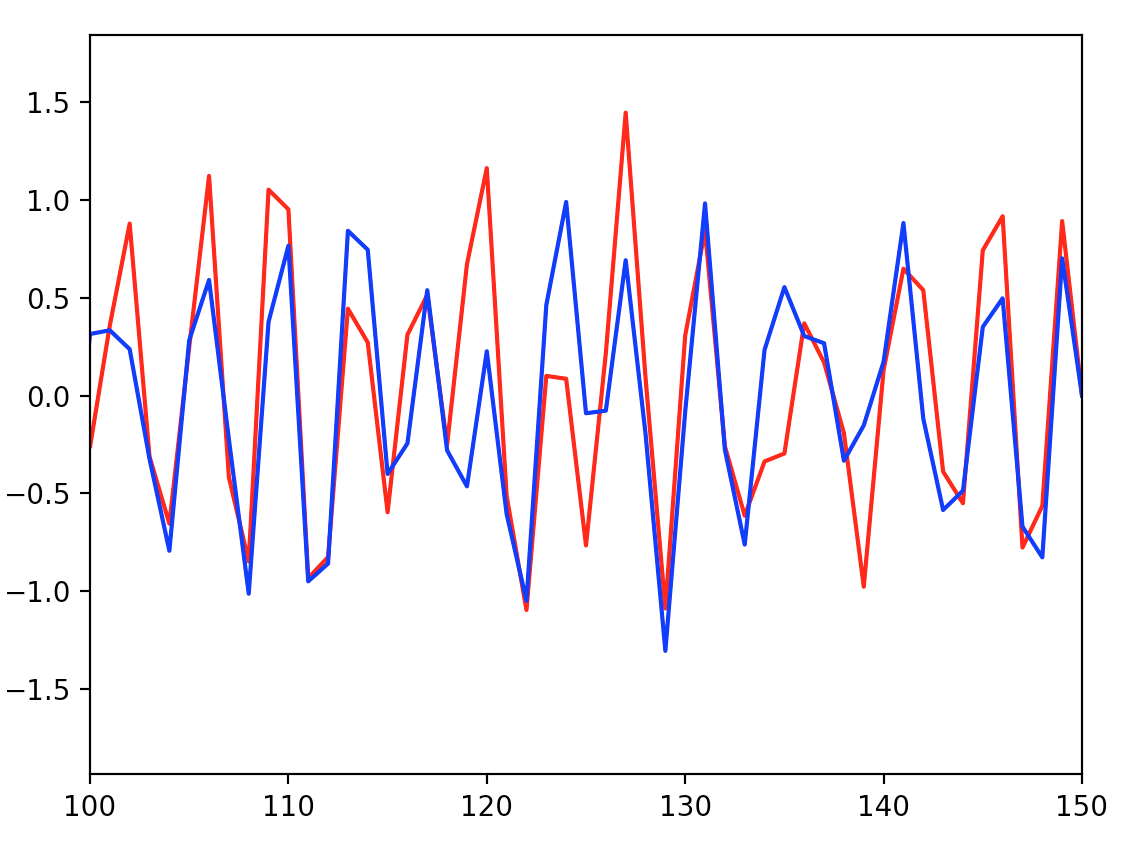
****

Figure3.2 optimal one-step linear predictor for r = 0.95

**<Analysis>**

In the Figure, the red one represent the original one and the blue one represent the estimated one. As you can see in the figures, when the parameter r is bigger, the blue one gets more closed to the red one, which means the effect of optimizing is better(the mean square is smaller). In the coding, we just plot the last 50 numbers of the whole process to make sure the generated samples are WSS, which looks more smoothly. Zn represent the estimated one and Yn is the original one. And definitely, the bigger r results in a more predictable process.