NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

Mobile Communication Systems (EE-451)

Homework 2 (CLO-1 & CLO-2)

Submission Details

Name	CMS ID
Muhammad Umer	345834
Submitted to:	Dr. Syed Ali Hassan
Class:	BEE-12
Semester:	7^{th}
Due:	26/10/2023

NUST
School of Electrical Engineering and Computer Science (SEECS)

Problem 1 (CLO-2):

Assume the receiver is located 10km from a 50W transmitter. The carrier frequency is 6GHz and the free space propagation is assumed, Gt=1 and Gr=1.

a) Evaluate the power at the receiver
b) Compute the magnitude of E-field at the receiver antenna
c) What is the receiver power in dBm?
d) If the receiver sensitivity is -96dBm, would the receiver be able to decode the message? Justify your answer.

Problem 2:

A brief measurement campaign indicates that the median propagation loss at 420 MHz in a mid-size city can be modeled by the following path loss equation

$$L = 25dB + 10log_{10}d^{2.8}$$

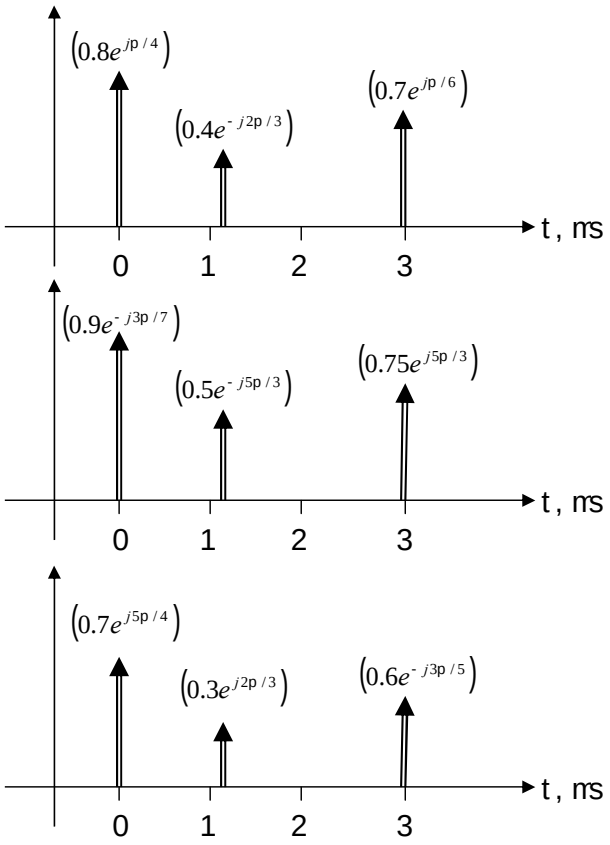
i.e., the path loss exponent is β = 2.8 and there is a 25 dB fixed loss.

a) Assuming a cell phone receiver sensitivity of -95 dBm, what transmitter power is required to service a circular area of radius 10 km?
b) Suppose the measurements were optimistic and β = 3.1 is more appropriate. What is the corresponding increase in transmit power (in decibels) that would be required?
c) If log-normal shadowing is present with σ = 8 dB, how much additional transmit power is required to ensure 10% thermal noise outage at a distance of 10 km?
d) For the same 10% probability of outage, estimate the fraction of total area within a radius of 10 km, where the signal is successfully decoded?

Problem 3 (CLO-2):

Suppose the three baseband channel impulse responses (IRs) on next page were measured when the transmitter was in a fixed position and the receiver (RX) was moved to three consecutive positions along a line, such that every pair of adjacent positions is separated by 2 wavelengths. Each IR corresponds to a distinct RX position.

a) Compute an estimate of the power delay profile for this channel.
b) Compute the rms delay spread for this channel.



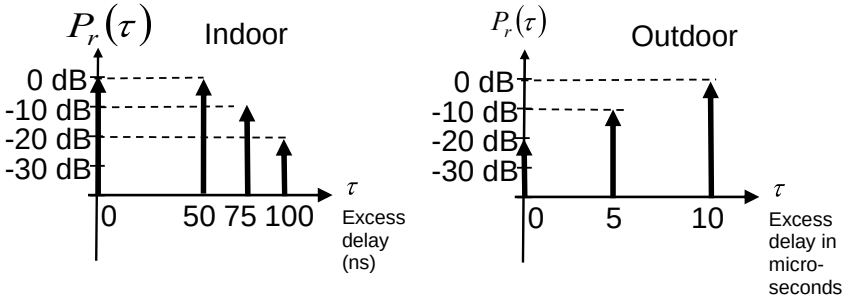
Problem 4 (CLO-2):

Let a static wireless channel be described by the following power delay profile: P (τ) = δ(τ) + 0.5 δ(τ-300ns). Suppose the transmitted signal is modulated with BPSK at a symbol rate of 10^5 symbols per second.

a) Compute the mean excess delay of the channel.
b) Evaluate the rms delay spread of the channel.
c) Classify this channel in terms of its type of fading.

Problem 5: Channels

a) Calculate the 90% and 50% correlation coherence bandwidths for the power delay profiles of following indoor and outdoor channels.



b) Would these channels be suitable for following standards without the use of an equalizer?

i.	GSM	bandwidth 200KHz
ii.	IS-95 (CDMA)	bandwidth 1.25MHz
iii.	WCDMA	bandwidth 5MHz
iv.	LTE	bandwidth 20MHz

Problem 6 (CLO-3):

Design and create a computer program that produces an arbitrary number of samples of propagation pathloss using a d^n pathloss model with lognormal shadowing. Your program is a radio propagation simulator, and should use, as inputs, the T-R separation, frequency, the pathloss exponent, the standard deviation of the log-normal shadowing, the close-in-reference distance, and the number of desired predicted samples. Your program should provide a check that ensures that the input T-R separation is equal to or exceeds the specified input close-in-reference distance and should provide a graphical output of the produced samples as a function of pathloss and distance (this is called a scatter plot).

Verify the accuracy of your computer program by running it for 50 samples at each of 5 different T-R separation distances (a total of 250 predicted pathloss values) and determine the best fit pathloss exponent and the standard deviation about the mean pathloss exponent of the predicted data using the techniques as described in example in the class. Draw the best fit mean pathloss model on the scatter plot to illustrate the fit of the model to the predicted values. You will know your simulator is working if the best fit pathloss model and the standard deviation for your simulated data is equal to the parameters you specified as inputs to your simulators.

Problem 7: Use MATLAB/Python (CLO-3):

a) Generate 200 samples of a Rayleigh random variable R with $E\{R^2\}=1$. Plot the samples in a stem plot. Remember that R = |X+jY|, where X and Y are zero mean, independent Gaussian random variables (r.v). Your Gaussian r.v. X and Y (produced by randn command) must each have equal variance equal to ½.

b) Give the estimated rms value of R, based on the 200 samples generated, which is given as $\sqrt{\overline{R^2}}$.

c) What fraction (if any) of these Rayleigh samples are 10dB below the estimated rms value? (Note that this threshold corresponds to $\rho = -10\text{dB}$ in the context of level crossing)

d) Generate 200 samples of a Rician random variable by adding means $m_r=5\cos(\pi/3)$ and $m_i = 5\sin(\pi/3)$, respectively to the real part (X) and imaginary part (Y) in part (a). Plot the samples in stem plot. What is the K factor of this Rician random variable?

e) Repeat part (d) except use $m_r=5\cos(\pi/6)$ and $m_i = 5\sin(\pi/6)$. Plot the samples in stem plot. What is the effect of phase change on the appearance of stem plot?

f) We now normalize the Rician random variables to have unit mean square value. Let Rn be the nth sample from Part (d). Make 200 normalized r.v. as $W_n = \frac{R_n}{\sqrt{\overline{R^2}}}$. Plot the W_n's as a stem plot and compare it to Part (a). What fraction of samples of W are 10dB below the rms value of W (should be fewer, because there should be less fading).

NUST
School of Electrical Engineering and Computer Science (SEECS)

Problem 6 (CLO-3):

Design and create a computer program that produces an arbitrary number of samples of propagation pathloss using a d^n pathloss model with lognormal shadowing. Your program is a radio propagation simulator, and should use, as inputs, the T-R separation, frequency, the pathloss exponent, the standard deviation of the log-normal shadowing, the close-in-reference distance, and the number of desired predicted samples. Your program should provide a check that ensures that the input T-R separation is equal to or exceeds the specified input close-in-reference distance and should provide a graphical output of the produced samples as a function of pathloss and distance (this is called a scatter plot).

Verify the accuracy of your computer program by running it for 50 samples at each of 5 different T-R separation distances (a total of 250 predicted pathloss values) and determine the best fit pathloss exponent and the standard deviation about the mean pathloss exponent of the predicted data using the techniques as described in example in the class. Draw the best fit mean pathloss model on the scatter plot to illustrate the fit of the model to the predicted values. You will know your simulator is working if the best fit pathloss model and the standard deviation for your simulated data is equal to the parameters you specified as inputs to your simulators.

The simulation baseline parameters are defined as follows:

```
distance_arr = np.array([50, 200, 500, 1000, 2000])
frequency = 10e6
d0 = 50  # close-in reference distance
PL_d0 = 20 * np.log10(4 * np.pi * d0 / (3e8 / frequency))
alpha = 3  # path loss exponent
N = 50  # number of samples
sigma = 4  # shadow fading standard deviation
lambda_ = 3e8 / frequency
```

To model a radio propagation model, we use the log-normal shadowing path-loss model, defined as:

$$PL(d) = PL(d_o) + 10 \times \alpha \log_{10}\left(\frac{d}{d_o}\right) + X_o$$

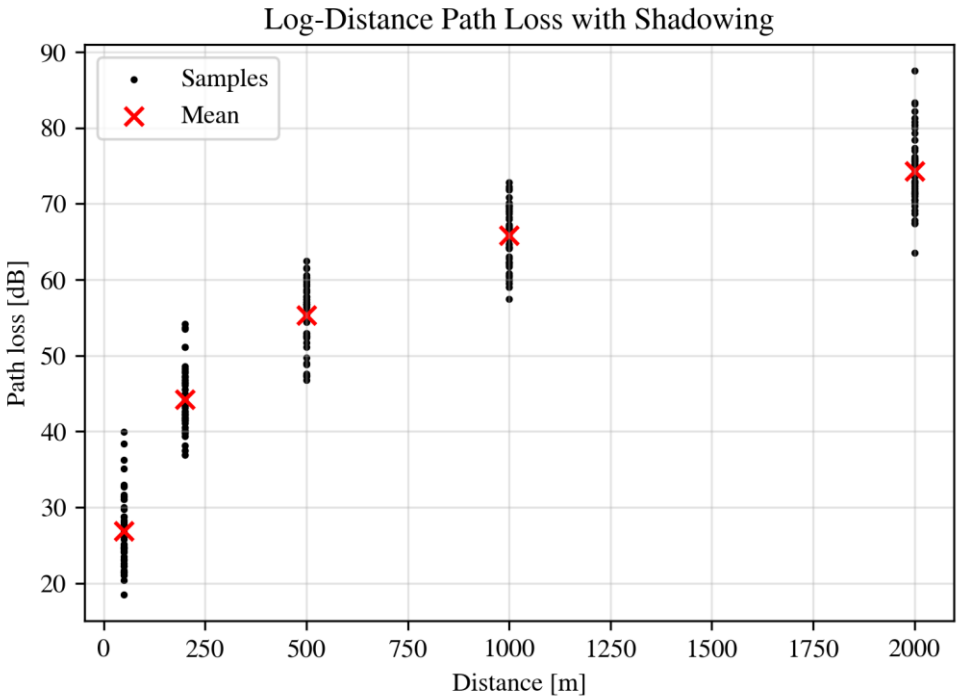
where X_o is the log-normal shadowing random variable (RV).

```
class PropagationSimulator:
    """A class to simulate path loss propagation."""

    def __init__(self, distance_arr, frequency, d0, alpha, N, sigma):
        """
        Constructs all the necessary attributes for the PropagationSimulator object.
        """
        self.distance_arr = distance_arr
        self.frequency = frequency
        self.alpha = alpha
        self.N = N
        self.sigma = sigma
```

```
        self.lambda_ = 3e8 / frequency
        self.d0 = d0
        self.d0_loss = 20 * np.log10(4 * np.pi * self.d0 / self.lambda_)

        # Required assertion as per question
        assert np.all(distance_arr >= d0), "All distances must be greater than
               or equal to d0."

        self.path_loss_arr = np.zeros((len(self.distance_arr), self.N))

        for i, distance in enumerate(self.distance_arr):
            for j in range(self.N):
                self.path_loss_arr[i, j] = self.log_distance(
                    distance, self.alpha, self.sigma
                )

    def log_distance(self, distance, alpha, sigma=None):
        """Log distance path loss model.

        Args:
            distance: Distance between transmitter and receiver.
            alpha: Path loss exponent.
            sigma: Shadow fading standard deviation.
        """

        loss = self.d0_loss + 10 * alpha * np.log10(distance / self.d0)
        if sigma is not None:
            loss += np.random.normal(0, sigma, np.shape(loss))

        return loss

    def plot_scatter(self):
        """Plots a scatter plot of the path loss."""

        plt.figure(figsize=(6, 4))
        plt.scatter(
            np.repeat(self.distance_arr, self.N), self.path_loss_arr, s=3, c="black"
        )
        plt.scatter(
            self.distance_arr,
            np.mean(self.path_loss_arr, axis=1),
            s=50, marker="x", c="red",
        )
        plt.xlabel("Distance [m]")
        plt.ylabel("Path loss [dB]")
        plt.legend(["Samples", "Mean"])
        plt.title("Log-Distance Path Loss")
        plt.grid(alpha=0.35)
        plt.show()
```

Then, using the defined function `plot_scatter()`. We obtain the following scatter plot:

```
simulator = PropagationSimulator(distance_arr, frequency, d0, alpha, N, sigma)
simulator.plot_scatter()
```



As per the example followed in class, we now define a cost function (MSE) and minimize it. The cost function is expressed as:

$$J(\alpha') = \sum_{i=0}^{4} (PL(d_i) - PL'(d_i; \alpha'))^2$$

where $PL'(d; \alpha') = PL(d_o) + 10 \times \alpha' \log_{10}\left(\frac{d}{d_o}\right)$ and is the estimator for the path-loss exponent, α'.

To find the extrema, we use ***sympy*** to symbolically differentiate the defined cost function as:

```
sym_alpha = sym.Symbol("alpha")

path_loss_ref = simulator.log_distance(distance_arr, alpha, sigma)
table = []
for i in range(len(distance_arr)):
    table.append([distance_arr[i], path_loss_ref[i]])
print(
    tabulate.tabulate(
        table, headers=["Distance [m]", "Path Loss [dB]"], tablefmt="fancy_grid"
    )
)

estimate = simulator.log_distance(distance_arr, sym_alpha)

def compute_MSE(path_loss_ref, estimate):
    mse = np.sum((path_loss_ref - estimate) ** 2)
    return mse

for i in range(len(distance_arr)):
    print(f"PL'({distance_arr[i]}m) = {estimate[i]}")
```

```
MSE = compute_MSE(path_loss_ref, estimate)
print(f"\nMSE = {sym.simplify(compute_MSE(path_loss_ref, estimate))}")

dMSE = sym.diff(MSE, sym_alpha) differentiation of MSE w.r.t. alpha
print(f"d_MSE/d_alpha = {sym.simplify(dMSE)}")

alpha_prime = sym.solve(dMSE, sym_alpha)[0]  # solve for alpha
print("\nPLE Estimate =", alpha_prime)
```

which yields the following outputs:

Distance [m]	Path Loss [dB]
50	27.86
200	40.46
500	63.40
1000	65.61
2000	71.05

```
PL'(50m) = 26.4211722727691
PL'(200m) = 6.02059991327962*alpha + 26.4211722727691
PL'(500m) = 10.0*alpha + 26.4211722727691
PL'(1000m) = 13.0102999566398*alpha + 26.4211722727691
PL'(2000m) = 16.0205999132796*alpha + 26.4211722727691

MSE = 562.1751498589*alpha**2 - 3358.29887913286*alpha + 5094.08909220518
d_MSE/d_alpha = 1124.3502997178*alpha - 3358.29887913286
```

Setting `d_MSE/d_alpha = 0` gives us the value of PLE estimate α'.

```
PLE Estimate = 2.98687951608654
```

Substituting α' back in cost function $J(\alpha')$, we can get the biased estimate of shadowing parameters σ as:

$$\sigma = \sqrt{\frac{J(\hat{\alpha})}{p}}$$

where p is the number of reference samples points.

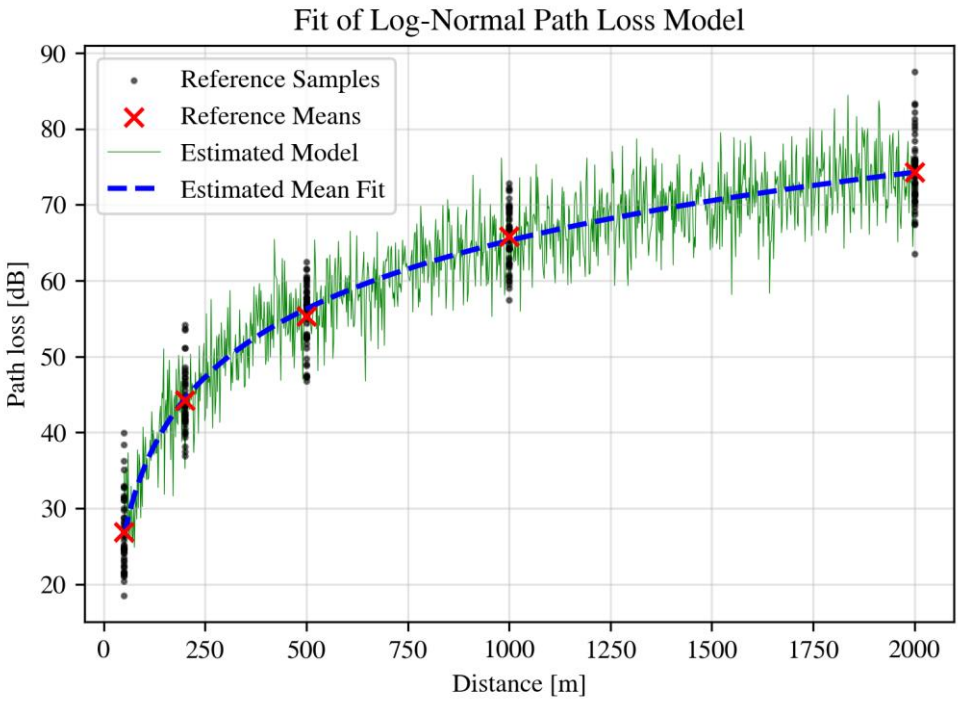
```
eval_MSE = MSE.evalf(subs={sym_alpha: alpha_prime})

sigma_sqrd_prime = eval_MSE / len(distance_arr)
sigma_prime = sym.sqrt(sigma_sqrd_prime)

print("Sigma Estimate =", sigma_prime)
```

which, finally, gives us the following estimate.

```
Sigma Estimate = 3.96666173793586
```

The obtained estimates closely approximate the input parameters, i.e., $\alpha = 3$ and $\sigma = 4$ dB. We can visualize the fitting characteristics of our approach by executing the following:

```
plt.figure(figsize=(6, 4))

plt.scatter(
    np.repeat(distance_arr, 50), simulator.path_loss_arr, s=3, alpha=0.5, c="black",
    zorder=10)
plt.scatter(distance_arr, np.mean(simulator.path_loss_arr, axis=1), s=50, marker="x",
    c="red", zorder=20)

d = np.linspace(d0, max(distance_arr), 1000)
est_fit = simulator.log_distance(d, alpha_prime, sigma_prime)
plt.plot(d, est_fit, "-", color="g", zorder=0, linewidth=0.35)
# estimation of log distance path loss reduces it to a non-random function
# i.e., E[X_o] = 0, where X_o is the shadow fading random variable
mean_est_fit = simulator.log_distance(d, alpha_prime)
plt.plot(d, mean_est_fit, "--", color="b", zorder=15, linewidth=2)

plt.xlabel("Distance [m]")
plt.ylabel("Path loss [dB]")
plt.legend(["Reference Samples", "Reference Means", "Estimated Model", "Estimated Mean
Fit"])
plt.grid(alpha=0.35)
plt.show()
```



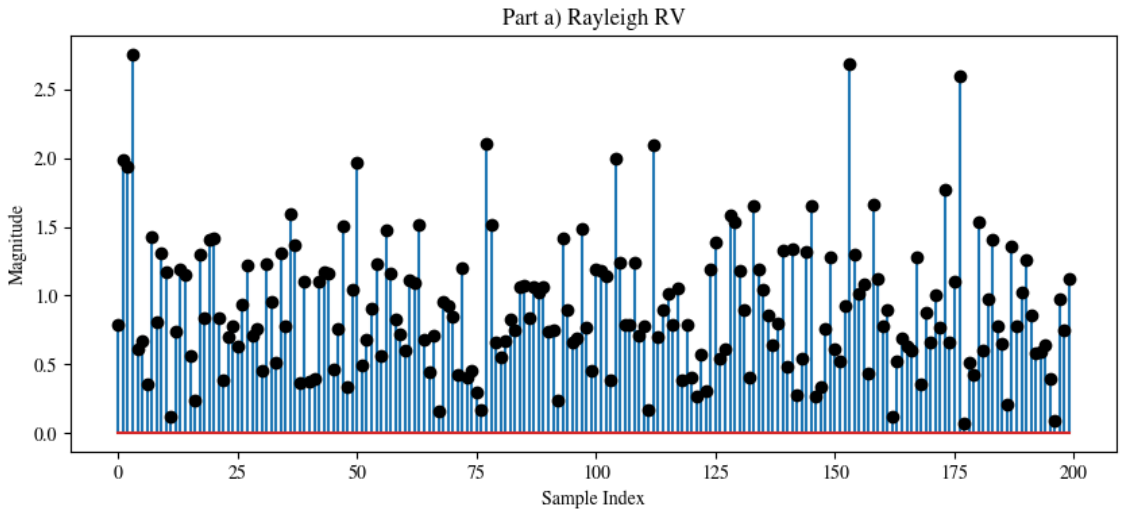
NUST
School of Electrical Engineering and Computer Science (SEECS)

Problem 7: Use MATLAB/Python (CLO-3):

a) Generate 200 samples of a Rayleigh random variable R with $E\{R^2\}=1$. Plot the samples in a stem plot. Remember that R = |X+jY|, where X and Y are zero mean, independent Gaussian random variables (r.v). Your Gaussian r.v. X and Y (produced by randn command) must each have equal variance equal to ½.

```
N = 200
var = 1 / 2
rv_rayleigh = np.random.normal(
    loc=0, scale=np.sqrt(var), size=N
) + 1j * np.random.normal(loc=0, scale=np.sqrt(var), size=N)
samples_rayleigh = np.abs(rv_rayleigh)

# Verify E{R^2}=1
print("Expectation of R^2: ", np.mean(samples_rayleigh**2))

# Stem plot
plt.figure(figsize=(10, 4))
plt.stem(samples_rayleigh, markerfmt="black")
plt.xlabel("Sample Index")
plt.ylabel("Magnitude")
plt.title("Part a) Rayleigh RV")
plt.show()
```

```
Expectation of R^2:  1.0318756271828502
```



b) Give the estimated rms value of R, based on the 200 samples generated, which is given as $\sqrt{\overline{R^2}}$.

```
rms_rayleigh = np.sqrt(np.mean(samples_rayleigh**2))
print("RMS value of R: ", rms_rayleigh)
```

```
RMS value of R:  1.0158127914054096
```

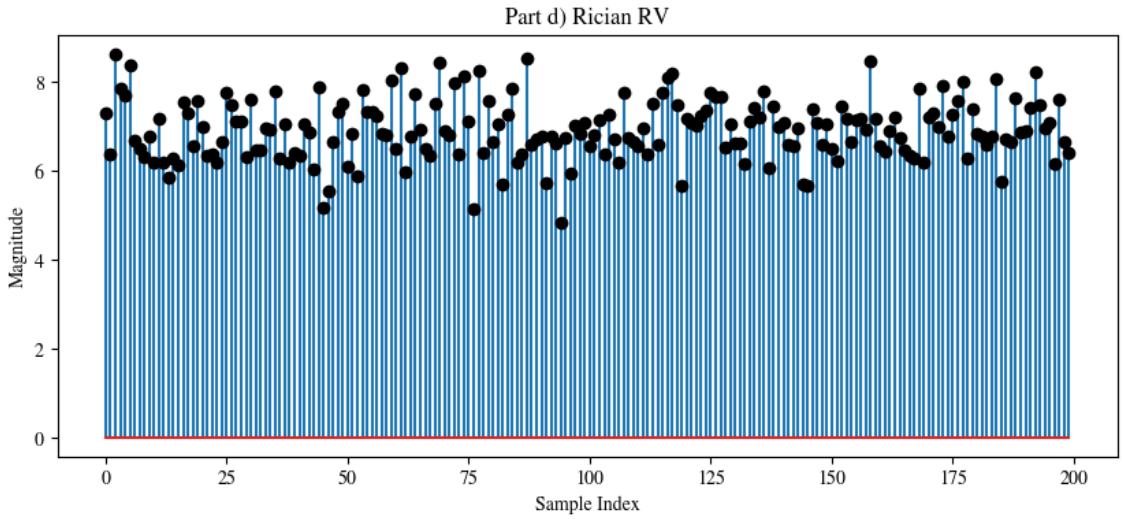
c) What fraction (if any) of these Rayleigh samples are 10dB below the estimated rms value? (Note that this threshold corresponds to $\rho = -10\text{dB}$ in the context of level crossing)

```
def pow2db(x):
    return 10 * np.log10(x)

fraction = np.sum(pow2db(samples_rayleigh) < pow2db(rms_rayleigh) - 10) / N
print("Fraction of Rayleigh Samples 10dB less than Estimated RMS: ", fraction)
```

```
Fraction of Rayleigh Samples 10dB less than Estimated RMS:  0.005
```

d) Generate 200 samples of a Rician random variable by adding means $m_r=5\cos(\pi/3)$ and $m_i = 5\sin(\pi/3)$, respectively to the real part (X) and imaginary part (Y) in part (a). Plot the samples in stem plot. What is the K factor of this Rician random variable?

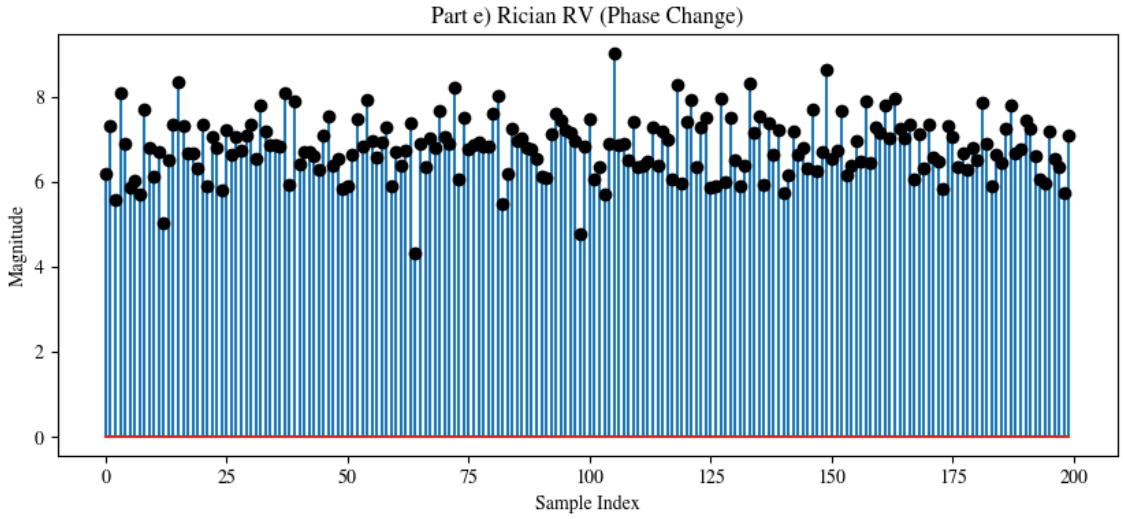
```
mr = 5 * math.cos(np.pi / 3)
mi = 5 * math.sin(np.pi / 3)
rv_rician = (mr + np.random.normal(loc=0, scale=np.sqrt(var), size=N)) + (
    1j * np.random.normal(loc=0, scale=np.sqrt(var), size=N) + mi
)
samples_rician = np.abs(rv_rician)

# Stem plot
plt.figure(figsize=(10, 4))
plt.stem(samples_rician, markerfmt="black")
plt.xlabel("Sample Index")
plt.ylabel("Magnitude")
plt.title("Rician Distribution Samples")
plt.show()

# K Factor
K = np.abs(mr + 1j * mi) ** 2 / (2 * np.var(samples_rician))
print("K Factor: ", K)
```




```
K Factor:  26.606118242668394
```

e) Repeat part (d) except use m_r=5cos(π/6) and m_i = 5sin(π/6). Plot the samples in stem plot. What is the effect of phase change on the appearance of stem plot?

```
mr = 5 * math.cos(np.pi / 6)
mi = 5 * math.sin(np.pi / 6)
rv_rician = (mr + np.random.normal(loc=0, scale=np.sqrt(var), size=N)) + (
    1j * np.random.normal(loc=0, scale=np.sqrt(var), size=N) + mi
)
samples_rician = np.abs(rv_rician)

# Stem plot
plt.figure(figsize=(10, 4))
plt.stem(samples_rician, markerfmt="black")
plt.xlabel("Sample Index")
plt.ylabel("Magnitude")
plt.title("Rician Distribution Samples")
plt.show()

# K Factor
K = np.abs(mr + 1j * mi) ** 2 / (2 * np.var(samples_rician))
print("K Factor: ", K, "dB")
```

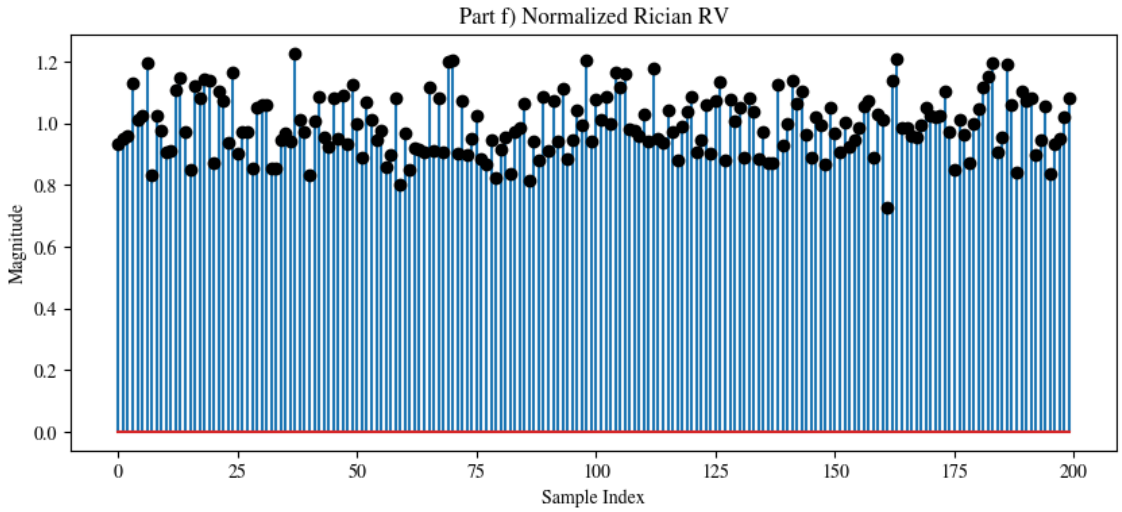



```
K Factor:  24.581052371559746
```

Change: There is no apparent change in the samples of the new Rician RV, as old $m_r = 5 \times \cos\frac{\pi}{3}$ is equal to the new $m_i' = 5 \times \sin\frac{\pi}{6}$, and similarly, old m_i is equal to the new m_r'. Hence, when plotting the magnitude of the samples of he RV, we observe no effect of the phase on the RV.

f) We now normalize the Rician random variables to have unit mean square value. Let Rn be the nth sample from Part (d). Make 200 normalized r.v. as $W_n = \frac{R_n}{\sqrt{R^2}}$. Plot the W_n's as a stem plot and compare it to Part (a). What fraction of samples of W are 10dB below the rms value of W (should be fewer, because there should be less fading).

```
mr = 5 * math.cos(np.pi / 3)
mi = 5 * math.sin(np.pi / 3)
rv_rician = (mr + np.random.normal(loc=0, scale=np.sqrt(var), size=N)) + (
    1j * np.random.normal(loc=0, scale=np.sqrt(var), size=N) + mi
)
samples_rician = np.abs(rv_rician)

Rn = samples_rician
Wn = Rn / np.sqrt(np.mean(Rn**2))

# Stem plot
plt.figure(figsize=(10, 4))
plt.stem(Wn, markerfmt="black")
plt.xlabel("Sample Index")
plt.ylabel("Magnitude")
plt.title("Rician Distribution Samples")
plt.show()

# RMS
rms = np.sqrt(np.mean(Wn**2))
print("RMS value of W: ", rms)

# Fraction of samples 10dB below RMS
fraction = np.sum(pow2db(Wn) < pow2db(np.sqrt(np.mean(Wn**2))) - 10) / N
print("Fraction of W Samples 10dB less than Estimated RMS: ", fraction)
```




```
RMS value of W:  1.0
Fraction of W Samples 10dB less than Estimated RMS:  0.0
```