ELE2035 Mathematics Coursework

Oliver Ross — 07/04/2025 — Student number: 40398890

Question 1

All parts of Question 1 require the code in Figure 1 to be run first to import the workspace variable x. The workspace variable file, along with the complete MATLAB script for this question, can be accessed here.

```
% importing the given MATLAB workspace file. Ensure the workspace file is in the
% same folder as this script
load('coursework_data_x.mat');
```

Figure 1: MATLAB code to import workspace variable x.

a)

By running the MATLAB script given in Figure 2 the sample mean $\bar{x}$, variance S^2, and skewness γ_1 of x are found to be

$$\bar{x} = 2.9182, \quad S^2 = 8.6164, \quad \gamma_1 = 2.0579.$$

Since the coefficient of skewness is positive, $\gamma_1 > 0$, the probability distribution of x is positively skewed. Furthermore, the magnitude of the skewness is high, indicating the probability distribution exhibits significant asymmetry with a much longer tail to the right of the probability distribution than to the left.

The random variable (RV) x has a large sample variance, $S^2 = 8.6164$, relative to the sample mean, $\bar{x} = 2.9182$. This indicates a higher probability that any given sample of x will fall further from the mean than if the variance were lower. This means that the samples of x are more spread out, indicating relatively high randomness.

```
% (a) finding mean, variance, and skewness of x.
mean_x = mean(x);
var_x = var(x);
skewness_x = skewness(x);
```

Figure 2: MATLAB code for finding the mean, variance, and skewness of x.

b)

By running the MATLAB script given in Figure 3, the probability that x belongs to the interval $[0, 1.0]$ is

$$P(0 \leq x \leq 1) = 0.2852.$$

```
% (b) P(0<=x<=1)
pb = (sum(x<=1)-sum(x<0))/length(x);
% counts how many samples have values that are less than or equal to 1, subtracts the
% number of samples with values less than zero, thus giving total number samples with
% values in interval [0, 1.0]. This number is then divided by total number of
% samples, giving the probability a sample value lies in this interval.
```

Figure 3: MATLAB code for finding the probability x belongs to the interval [0, 1.0].

c)

Running the MATLAB script given in Figure 4 generates an approximation of the probability density function (PDF) of x, which is shown in Figure 5.

```
% (c) plotting PDF of x
binwidth = 0.1; % defining interval widths for the histogram
xRange = [-5:binwidth:20]; % defining points for histogram x-axis
N = hist(x, xRange); % counts the number of x samples with values in each bin

figure(Name='Q1(c)'); % generate a figure just showing the PDF of x over x
% plotting the probability of an x sample being in each bin over x to give
% a good approximation of the PDF of x:
plot(xRange, N./(binwidth*length(x)), LineWidth=1.5);
% improving graph readability:
xlabel('x')
ylabel('PDF')
title('PDF of x Plotted Over x')
```

Figure 4: MATLAB code to plot the PDF of x.

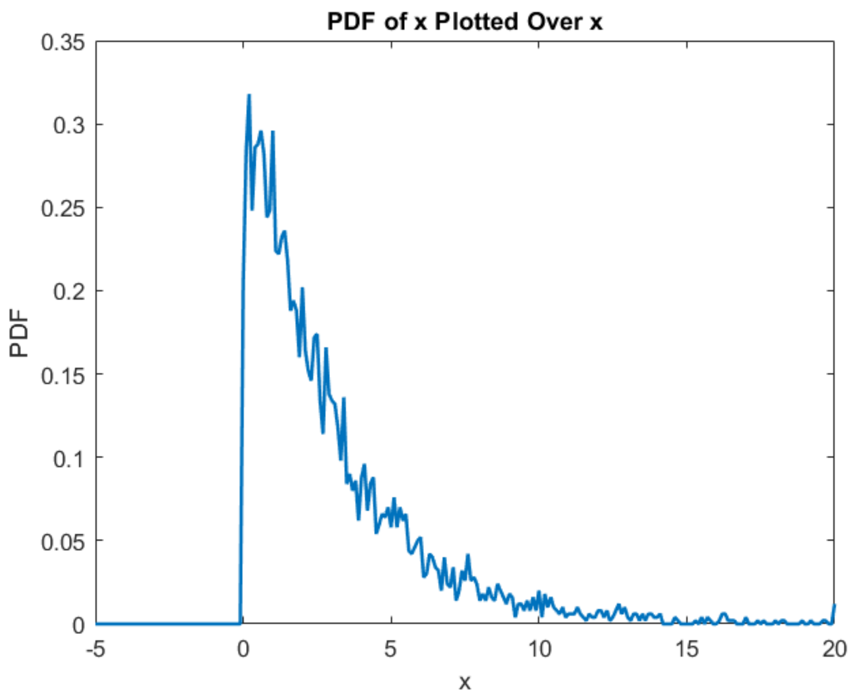

Figure 5: Approximation of the PDF of x generated in MATLAB for binwidth = 0.1.

d)

Running the MATLAB script given in Figure 6 generates a plot showing an approximation of the probability density function (PDF) of x on the same axes as the PDF for an exponentially distributed RV of the same mean as x, which is shown in Figure 7.

```
% (d) comparing PDF of x with PDF of equal mean exponentially distributed RV
binwidth = 0.1; % defining interval widths for the histogram
xRange = [-5:binwidth:20]; % defining points for histogram x-axis
N = hist(x, xRange); % counts the number of x samples with values in each bin

figure(Name='Q1(d)'); % generate a separate figure for comparing PDF of x to exponential distribution
hold on
plot(xRange, N./(binwidth*length(x)), LineWidth=1.5); % plot PDF of x again
% generate the PDF of an exponentially distributed RV with the same mean as x, evaluated at the same
% points as the PDF of x:
expRV_pdf = exppdf(xRange, mean(x));
% plot the PDF of the exponentially distributed RV on the same axes as the PDF of x:
plot(xRange, expRV_pdf, LineWidth=2, LineStyle="--");
% improving readability of the plot
legend('PDF of x','PDF of exponentially distributed RV')
title('Comparing PDF of x with PDF of Exponentially Distributed RV of Equal Mean')
xlabel('x')
ylabel('PDF')
```

Figure 6: MATLAB code to plot the PDF of x on the same axes as the PDF of an exponentially distributed RV with the same mean as x.

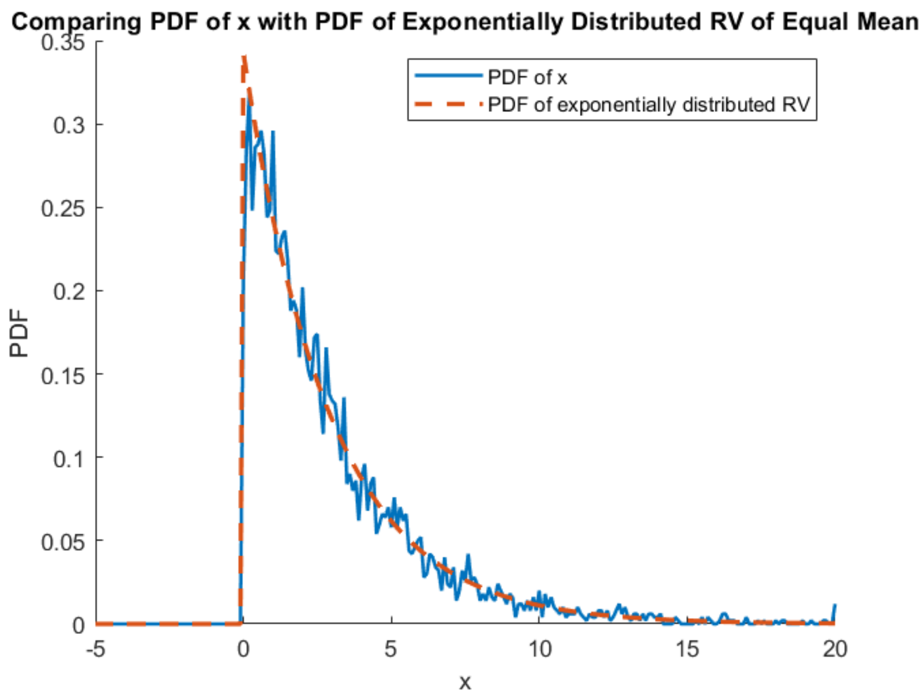

Figure 7: Approximation of the PDF of x plotted on the same axes as the PDF of an exponentially distributed RV with the same mean as x generated in MATLAB.

Figure 7 shows that the PDF of x closely resembles the PDF of the exponentially distributed RV. Therefore, it is likely that the RV x also follows an exponential distribution with $\lambda = 1/\bar{x} = 1/2.9182 = 0.3427$, summarised as $x \sim E(0.3427)$; the small variations between the PDF of x and the PDF of the exponentially distributed RV with the same mean are caused by randomness introduced by sampling x. If this conclusion is true, it is likely that if more samples of x are taken the PDF of x will more closely resemble the exponential distribution, and if infinite samples are taken the PDF of x would match the exponential distribution exactly.

e)

Running the MATLAB script given in Figure 8 generates a scatter plot of x and y for each sample, which is shown in Figure 9. Furthermore, running this code also allows the covariance $\hat{\sigma}_{xy}$ and correlation coefficient $\hat{\rho}_{xy}$ of x and y to be found as

$$\hat{\sigma}_{xy} = 8.6445, \quad \hat{\rho}_{xy} = 0.8218.$$

```
% (e) Checking correlation between x and y
load("coursework_data_y.mat"); % ensure file is in same folder as script
% creating a scatter plot of y over x:
figure(Name='Q1(e)'); % new figure to show the scatter plot
hold on
scatter(x, y); % creating scatter plot with x on x-axis and y on y-axis
% improving readability of the plot
xlabel('x');
ylabel('y');
title('Scatter Plot to Show Correlation Between y and x');
lsline; % adding a line of best fit
legend('Sample data', 'Best fit line', Location='southeast')
% finding covariance and correlation coefficent
covariance_xy = cov(x, y);
rho_xy = corrcoef(x, y);
```

Figure 8: MATLAB code to generate a scatter plot of y over x for each sample point.

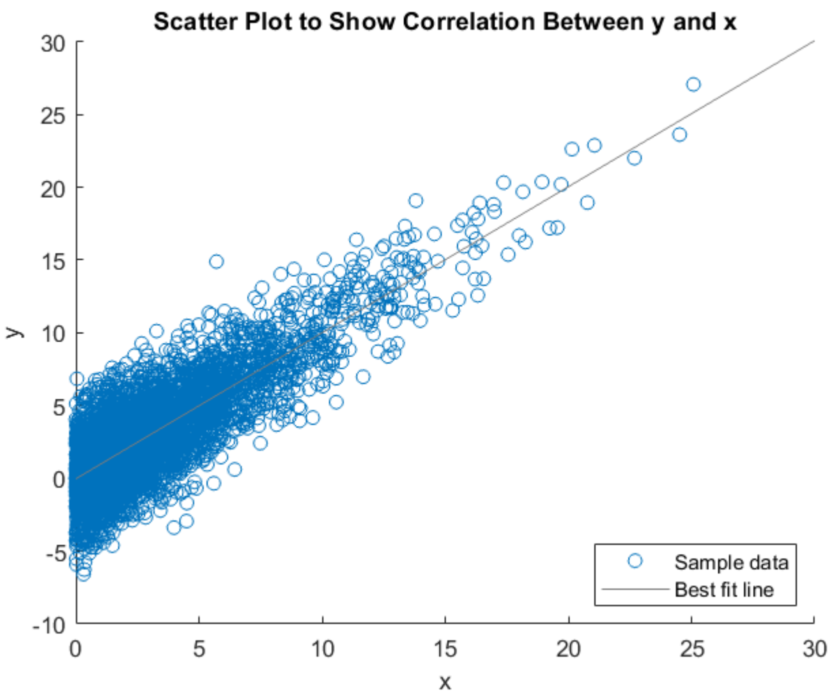

Figure 9: Scatter plot of y over x generated in MATLAB.

From Figure 9 we can see that x and y have a strong positive correlation. This is consolidated by the calculated correlation coefficient of $\hat{\rho}_{xy} = 0.8218$, where a correlation coefficient of 1 indicates perfect positive correlation, 0 indicates no correlation, and -1 indicates perfect negative correlation.

Question 2

The complete MATLAB script for this question can be accessed here.

a)

Running the code given Figure 10, 100000 realisations of the achievable rate R can be generated.

```
% (a) generating 100 000 realisations of R
P = 10; % constant given in question
M = 2; % constant given in question
% creating empty 100000 element column vector to hold R values:
R_M2 = zeros([100000 1]);
for realisationNum = 1:100000 % repeat this loop 100 000 times
   % creating independent standard normal RVs hm1 and hm2
   hm1 = randn([M 1]);
   hm2 = randn([M 1]);
   % defining the vector h
   h = (1 / sqrt(2))*hm1+(sqrt(-1)/sqrt(2))*hm2;
   norm_h = norm(h); % find the norm/length of h
   % add the R value calculated for this iteration to the R array:
   R_M2(realisationNum, 1) = log2(1+P*(norm_h)^2);
end
clear realisationNum % remove indexing variable from workspace
```

Figure 10: MATLAB code which generates 100000 realisations of achievable rate R for a multiple-input single-output (MISO) system when $M = 2$.

b)

Executing the code given in Figure 11 after the code in Figure 10, the mean value of the achievable rate is

$$\bar{R} = 4.0581,$$

although due to the randomness of the independent standard normal RVs $h_{m,1}$ and $h_{m,2}$, this mean is subject to small changes each time the MATLAB script is run.

```
% (b) finding the mean value of R
mean_R_M2 = mean(R_M2);
```

Figure 11: MATLAB code to find the mean value of achievable rate when $M = 2$.

c)

The code in Figure 12 generates the cumulative distribution function (CDF) of R for $M = 2$ shown in Figure 13.

```
% (c) plotting the CDF of R
figure(Name='Q2(c)'); % individual plot to show CDF when M = 2
cdfplot(R_M2) % plot the CDF of R when M=5
% improving plot readability
xlabel('R (bits/s/Hz)')
ylabel('CDF')
title('CDF of Achievable Rate R Plotted Over R for M=2')
```

Figure 12: MATLAB code to plot the CDF of R when $M = 2$.

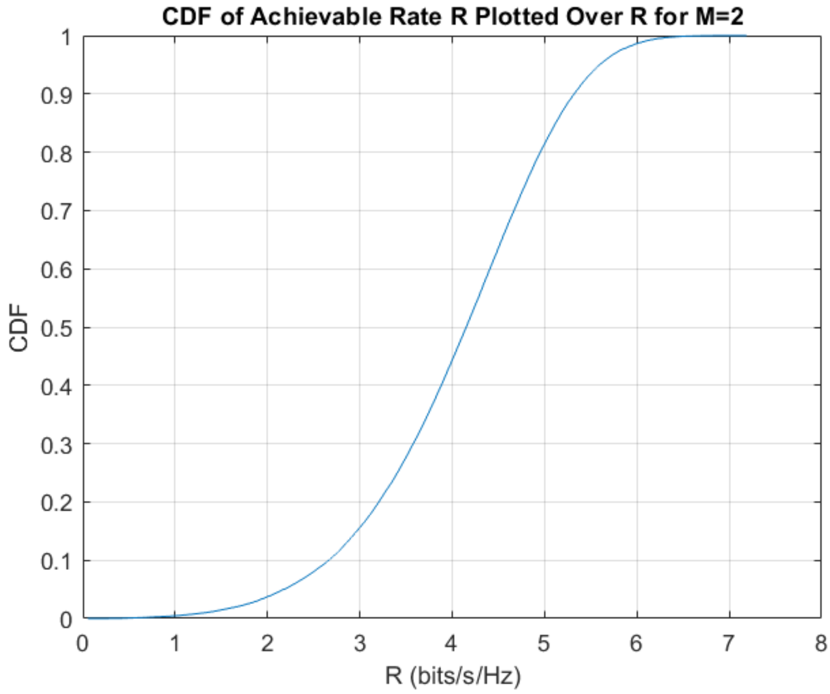

Figure 13: CDF of R plotted over R when $M = 2$ generated using MATLAB.

d)

Given that system outage occurs when $R < 1$, the outage probability can be found in MATLAB by running the code given in Figure 14,

$$\text{outage probability} = P(R < 1) = 0.0047.$$

The randomness of the independent standard normal RVs $h_{m,1}$ and $h_{m,2}$ means this probability is subject to small changes each time the script is run. The outage probability can typically be rounded to 0.5% ($P(R < 1) = 0.005$). The chance of outage is very small since R is rarely less than 1, so the system is well designed. Despite this, reduction in outage probability is still desirable to further improve the system.

```
% (d) Finding system outage probabilty, P(R<1)
outage_probability_M2 = sum(R_M2<1)/length(R_M2);
% find the total number of R values less than 1, and divide by total number
% of R values to give the probability that R is less than 1
```

Figure 14: MATLAB code to determine the outage probability of the system when $M = 2$.

e)

The code in Figure 15 generates the CDF of R for $M = 50$ shown in Figure 16.

```
% (e) repeating with M = 50 value
P = 10; % constant given in question
M = 50; % constant given in question
% creating empty 100000 element column vector to hold R values:
R_M50 = zeros([100000 1]);
for realisationNum = 1:100000 % repeat this loop 100 000 times
    % creating independent standard normal RVs hm1 and hm2
    hm1 = randn([M 1]);
    hm2 = randn([M 1]);
    % defining the vector h
    h = (1 / sqrt(2))*hm1+(sqrt(-1)/sqrt(2))*hm2;
    norm_h = norm(h); % find the norm/length of h
    % add the R value calculated for this iteration to the R array:
    R_M50(realisationNum, 1) = log2(1+P*(norm_h)^2);
end
clear realisationNum % remove indexing variable from workspace
figure(Name='Q2(e)'); % individual plot to show CDF when M = 50
cdfplot(R_M50); % plot the CDF of R when M=50
% improving plot readability
xlabel('R (bits/s/Hz)'); ylabel('CDF'); title('CDF of Achievable Rate R Plotted Over R for M=50');
```

Figure 15: MATLAB code to plot the CDF of R when $M = 50$.

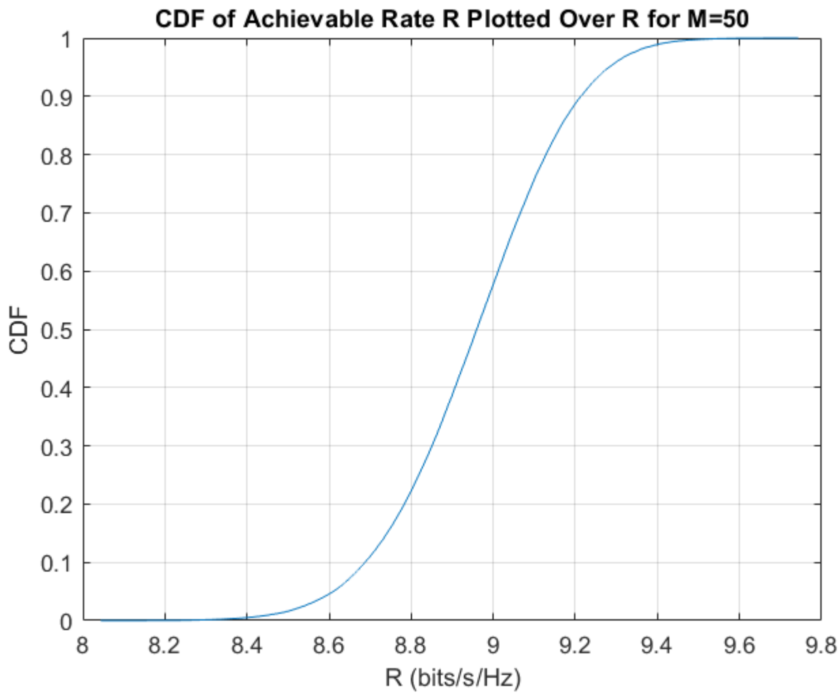

Figure 16: CDF of R plotted over R when $M = 50$ generated using MATLAB.

Comparing Figures 13 and 16, we can see that for both $M = 2$ and $M = 50$ the shape of the CDF is characteristic of a normally distributed RV, so the instantaneous achievable rate seems to be normally distributed regardless of the number of transmit antennas M. Running the MATLAB code in Figure 17, the mean achievable rate and outage probability when $M = 50$ are

$$\bar{R} = 8.9543, \quad P(R < 1) = 0,$$

although due to the randomness of the independent standard normal RVs $h_{m,1}$ and $h_{m,2}$, these values are subject to small changes each time the MATLAB script is run.

The mean achievable rate when $M = 50$ is much larger than the mean achievable rate when $M = 2$, allowing the conclusion to be drawn that increasing the number of transmit antennas M makes a higher achievable rate R more probable. Furthermore, comparing Figures 13 and 16, we can see that when $M = 50$ the CDF curve is much steeper than when $M = 2$, meaning that there is less dispersion (lower variance) in the achievable rate when $M = 50$, making the system with more transmit antennas more predictable and therefore more reliable. This also makes it much less likely that the system enters outage, since a higher mean and lower variability makes it much less likely that the instantaneous achievable rate falls beneath $R = 1$.

```
% finding the mean and outage probability for M = 50 for comparison
% with M=2:
mean_R_M50 = mean(R_M50);
outage_probability_M50 = sum(R_M50<1)/length(R_M50);
```

Figure 17: MATLAB code to find the mean achievable rate and outage probability when $M = 50$.