Lab 1: Random variables and estimation theory

Statistical Signal Processing (5CTA0) - Lab Assignment

Lecturers: Simona Turco (Flux 7.076)

Assistants: Yizhou Huang (Flux 7.078), Agata Barbagini (Flux 7.079), Xueting Li (Flux 7.079), Luan Fiorio (Flux 7.066)

Students (Name + IDs): Wei Zheng + 1886061 Yuncheng Yuan + 1778307 Sizhuo Yao + 1895591

Group #: Lab Group 6

General info and guidelines

To complete and submit the Labs:

•All students must register on Canvas.

•Each lab will be carried out in groups of 3 students, randomly assigned on Canvas.

•All information and data will be available on Canvas.

•Each group must carry out all assignments of the Lab and hand in a report for each Lab.

•The report is obtained by exporting this matlab live script as pdf. This is the only file that you need to provide.

•Sometimes, ruuning the code within livescript might be slow. You may consider to first implment and test your code in a seperate .m file, and copy the code to livescript in the end.

•The report must be submitted through Canvas.

•In case of problems, the labs can alternatively be submitted by printing them and putting them in the post box of one of the teaching assistants (Flux floor 7th, opposite to the secretary), together with an email notification.

•The report must be accompanied by the peer-review form, which is used to provide an indication of the contribution of each student in the group. The link to the peer review form is available on Canvas.

•If plagiarism is detected, the Lab will be judged with 0 points.

Below, some useful tips for working with MATLAB and for producing a neat Lab report:

•Make use of comments to divide the code in sections and facilitate its understanding.

•Use the command `doc or help` to learn how to use a specific MATLAB function (e.g.`doc stem`).

•Make sure that all your graphs are easily readable, even when printed in black and white. Use for this purpose the options of the plot/stem function to properly increase the dimension and/or change the marker symbols, for example:

```
stem(x,y, 'MarkerSize',10)
plot(x,y, 'LineStyle',':','Color','r','Marker','*')
```

•Make sure that all your graphs have their axes properly labeled and a legend when several signals are plotted on the same graph. Also, make sure the all the text is readable. Below some useful commands for these purposes:

```
xlabel('lag n', 'FontSize', 14, 'FontWeight', 'Bold')
ylabel('r[n]', 'FontSize', 14, 'FontWeight','Bold')
legend('r_x[n]', 'r_x_y[n]')
```

Credits and deadlines

Each Lab counts for 15% of the final grade, for a total of 30%. The remaining 70% is determined by the final written exam.

Please submit this lab on Canvas by the 19th March at 23.59.

Part 1 - Introduction to random signals

Assignment 1 - Sampling of a random variable

In the following experiment, we construct a random variable X_N as the normalized sum of N RVs X_i as follows:

$$X_N = \frac{1}{N}\sum_{i=1}^{N} X_i \text{ with } N = 1, 2, \ldots$$

The RVs X_i, for $i = 1, 2, \ldots, N$ are N IID stationary RVs with mean $E[X_i] = \mu_i = \mu$ and variance $E\left[(X_i - E[X_i])^2\right] = \sigma_i^2 = \sigma^2$.

a) Calculate an analytical expression for the mean $\mu_{X_N} = E[X_N]$and the variance $\sigma^2_{X_N} = E\left[X_N^2\right] - E[X_N]^2$ of the RV X_N as a function of N.

Hint: The X_i are IID (independent, identically distributed)

ANSWER:

1.The mean of the sum of independent random variables is equal to the sum of their means:

$$E[X_N] = \frac{1}{N} * \sum_{i=1}^{N} E[X_i] = \frac{1}{N} * N * \mu = \mu$$

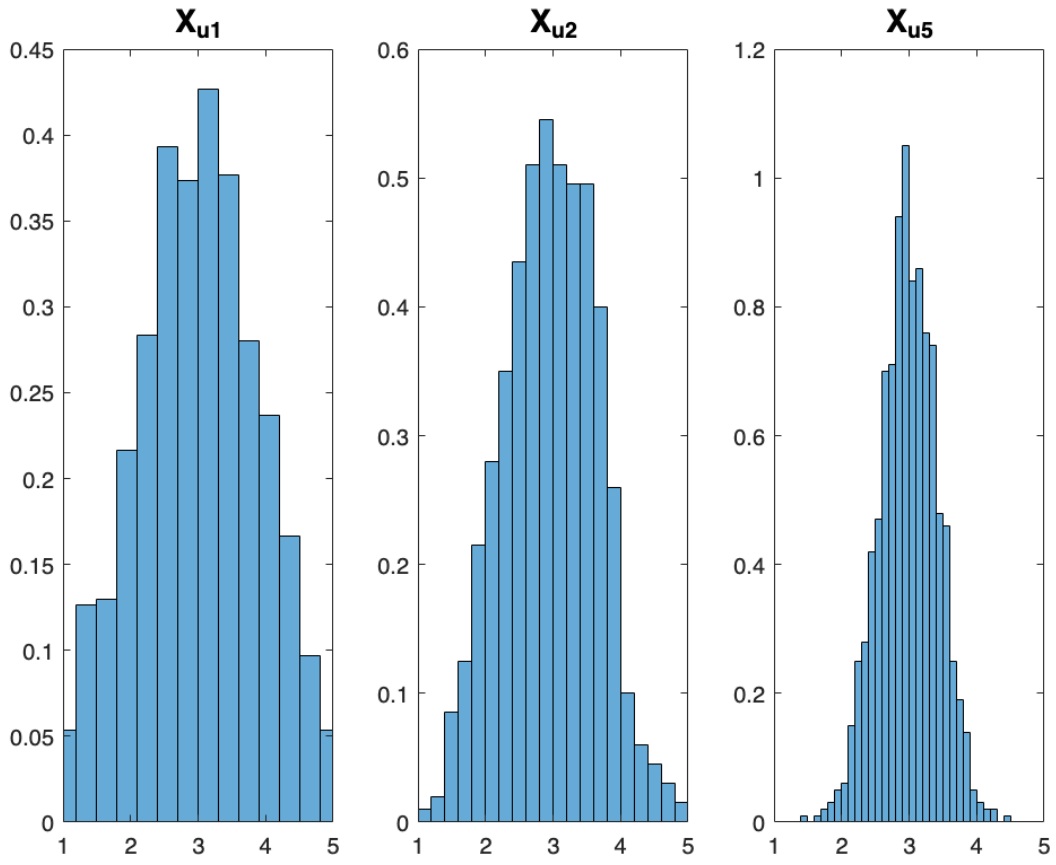
2. and the variance:

$$\sigma^2_{X_N} = E\left[(X_N - E[X_N])^2\right] = E\left[\left(\frac{1}{N}\sum_{i=1}^{N} X_i - E\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right]\right)^2\right] = \frac{1}{N^2}\sum_{i=1}^{N} E\left[(X_i - E[X_i])^2\right] = \frac{1}{N^2} * N * \sigma^2 = \frac{\sigma^2}{N}$$

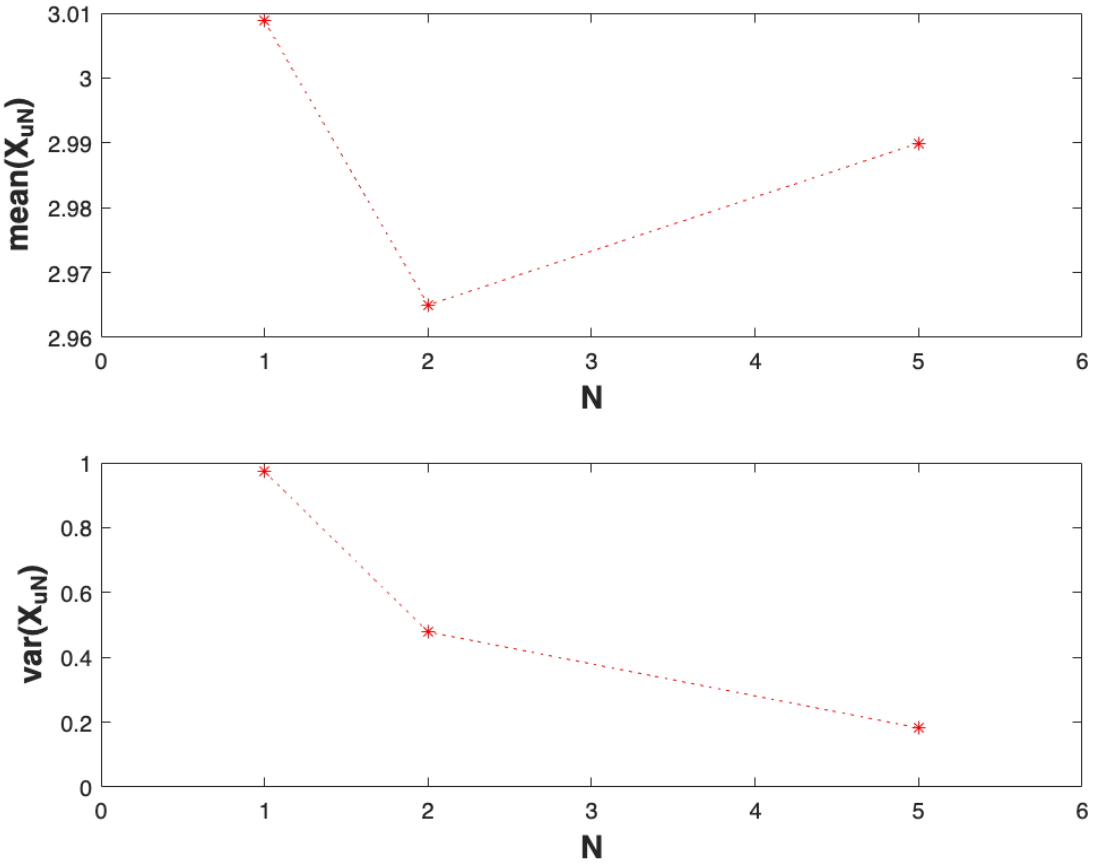
b) Implement a Monte-Carlo simulation to generate RVs X_i, for $i = 1, 2, \ldots 5$, with gaussian distribution. Generate for each of these RVs 1000 IID samples with parameters of the gaussian distribution $\mu_{x_i} = \mu_x = 3$ and $\sigma_{x_i} = \sigma_x = 1$. Construct X_{uN} for $N = 1, 2$ and $N = 5$ and make a subplot of X_{u1}, X_{u2} and X_{u5}. (plot the normalized histograms as a representation of the PDF). Please make sure that the x-axis is the same for all subplots. Additionally, make two subplots showing how the mean and variance of X_N change with N.

Hint: Use MATLAB functions rand, mean, var, and subplot. Set same axis limit for the three subplots.

```
clear all
close all
clc

%--------------------------------------------------
% Monte Carlo simulation
%--------------------------------------------------
% your code goes here (~10 lines of code, excluding the plotting code)
mu_x = 3; sigma_x = 1;
N = [1, 2, 5];
X = cell(1, 5);

% Generate RVs X_i with gaussian distribution
for i = 1:5
    X{i} = normrnd(mu_x, sigma_x, 1, 1000);
end

% Construct X_uN for N=1,2,5
X_un = cell(1, 3);
for i = 1:length(N)
    n = N(i);
    X_un{i} = mean(reshape([X{1:n}], [], n), 2);
end

% Plot the histograms of X_un
figure;
for i = 1:3
    subplot(1, 3, i);
    histogram(X_un{i}, 'Normalization', 'pdf');
    xlim([1, 5]);
    title(['X_{u', num2str(N(i)), '}'], 'FontSize', 14, 'FontWeight','Bold');
end
```



```
% Calculate and plot the mean and variance of X_un
figure;
subplot(2, 1, 1);
plot(N, cellfun(@mean, X_un), 'LineStyle',':','Color','r','Marker','*');
xlabel('N', 'FontSize', 14, 'FontWeight', 'Bold'); ylabel('mean(X_{uN})', 'FontSize', 14, 'FontWeight', 'Bold');
xlim([0, 6]);
subplot(2, 1, 2);
plot(N, cellfun(@var, X_un), 'LineStyle',':','Color','r','Marker','*');
xlabel('N', 'FontSize', 14, 'FontWeight', 'Bold'); ylabel('var(X_{uN})', 'FontSize', 14, 'FontWeight', 'Bold');
xlim([0, 6]);
```



c) Do the plots of assignments (b) follow the analytical results that you found in answer (a)? Can you give a general statement about the mean and variance of a RV which consists of the normalized sum of N RVs with same distribution and finite mean and variance?

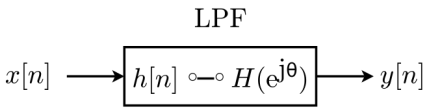
ANSWER:

Yes, the plots of assignments (b) follow the analytical results found in answer (a). As N increases, the distribution of X_{uN} becomes narrower, which corresponds to the decrease in variance as shown in the second subplot of the figure. The mean remains constant at $\mu = 3$ for all values of N, as expected from the analytical result.

In general, the mean of a RV which consists of the normalized sum of RVs with the same distribution and finite mean is equal to the mean of the original distribution, while the variance is inversely proportional to the number of RVs in the sum.

Assignment 2 - Autocorrelation and power spectral density

In this section, we will filter an input signal, which is a discrete random process, to generate an output signal, which is a new discrete random processes. We will study the statistical properties of the output process and its relation with the input process and the filtering operation. The figure below a system in which the input signal $x[n]$ is a white Gaussian random process with zero mean and unit variance. This input signal $x[n]$ is used as the input to a Low Pass Filter (LPF), from which the impulse response is denoted by $h[n]$ and the frequency response by $H(e^{j\theta})$.



In first instance we are using the very simple LPF:

$$h[n] = \frac{1}{2}(\delta[n] + \delta[n-1] - \delta[n-2])$$

Thus, in this case the output random process y[n] can be described by the following difference equation:

$$y[n] = \frac{1}{2}(x[n] + x[n-1] - x[n-2])$$

which implies that the output samples are averaged values of 3 succeeding input samples.

a) Derive by hand the autocorrelation $r_y[l] = E\{y[n]y[n-l]\}$ of the random process $y[n]$.

ANSWER:

$r_y[l] = E\{y[n]y[n-l]\}$

$r_y[l] = E\left\{\left[\frac{1}{2}(x[n] + x[n-1] - x[n-2])\right] * \left[\frac{1}{2}(x[n-l] + x[n-l-1] - x[n-l-2])\right]\right\}$

$r_y[l] = E\left\{\frac{1}{4}\left[x[n]x[n-l] + x[n]x[n-l-1] - x[n]x[n-l-2] + x[n-1]x[n-l] + x[n-1]x[n-l-1] - x[n-2]x[n-l-2]\right]\right\}$

$r_y[l] = \frac{1}{4}(3r_x[l] + 2r_x[l+1] + r_x[l+2] + 2r_x[l-1] + r_x[l+2])$

$r_y[l] = \frac{1}{4}(3r_x[l] + 2r_x[l+1] + r_x[l+2] + 2r_x[l-1] + r_x[l+2])$

Because the $\sigma_x{}^2 = 1 = r_x[l]$, mean value of distrubution is 0. Therefore, when l=0, $r_x[l] = 0$. Now we can get the result of autocorrelation of $y[n]$:

$r_y[l] = \frac{1}{4}[3\delta[l] + 2\delta[l+1] + \delta[l+2] + 2\delta[l-1] + \delta[l+2]]$

b) Derive by hand the PSD $P_y(e^{j\theta})$ of the random process $y[n]$ in two different ways, namely first as the FTD of $r_y[l]$ and second as $P_y(e^{j\theta}) = P_x(e^{j\theta}) \cdot |H(e^{j\theta})|^2$. Please simplify your final results as a function of 'cos' or 'sin' with the help of Euler's formula.

ANSWER:

$P_y[l] = \frac{1}{4}[3\delta[l] + 2\delta[l+1] + \delta[l+2] + 2\delta[l-1] + \delta[l+2]]$

$P_y(e^{j\theta}) = \frac{1}{4}\sum[3\delta[l] + 2\delta[l+1] + \delta[l+2] + 2\delta[l-1] + \delta[l+2]e^{j\theta}]$

First Result: $P_y(e^{j\theta}) = \frac{3}{4} + \frac{1}{2}P_x(e^{j\theta}) + \frac{1}{4}P_x(e^{j2\theta}) + \frac{1}{2}P_x(e^{-j\theta}) + \frac{1}{4}P_x(e^{-j2\theta}) = \frac{3}{4} + \cos\theta + \frac{1}{2}\cos 2\theta$

Because $P_y(e^{j\theta}) = P_x(e^{j\theta}) \cdot |H(e^{j\theta})|^2$, therefore, we can get:

$h[n] = \frac{1}{2}(\delta[n] + \delta[n-1] - \delta[n-2])$

$H(e^{j\theta}) = \frac{1}{2}(1 + e^{-j\theta} + e^{-j2\theta})$

$|H(e^{j\theta})|^2 = H(e^{j\theta}) * H(e^{j\theta}) = \frac{1}{4}(1 + e^{-j\theta} + e^{-j2\theta}) * (1 + e^{-j\theta} + e^{-j2\theta})$

$|H(e^{j\theta})|^2 = \frac{1}{4}(3 + 2e^{-j\theta} + 2e^{j\theta} + e^{-j2\theta} + e^{j2\theta})$

$|H(e^{j\theta})|^2 = \frac{3}{4} + \cos\theta + \frac{1}{2}\cos 2\theta$

$P_x(e^{j\theta}) = \sigma_x{}^2 = 1$

Second Result:

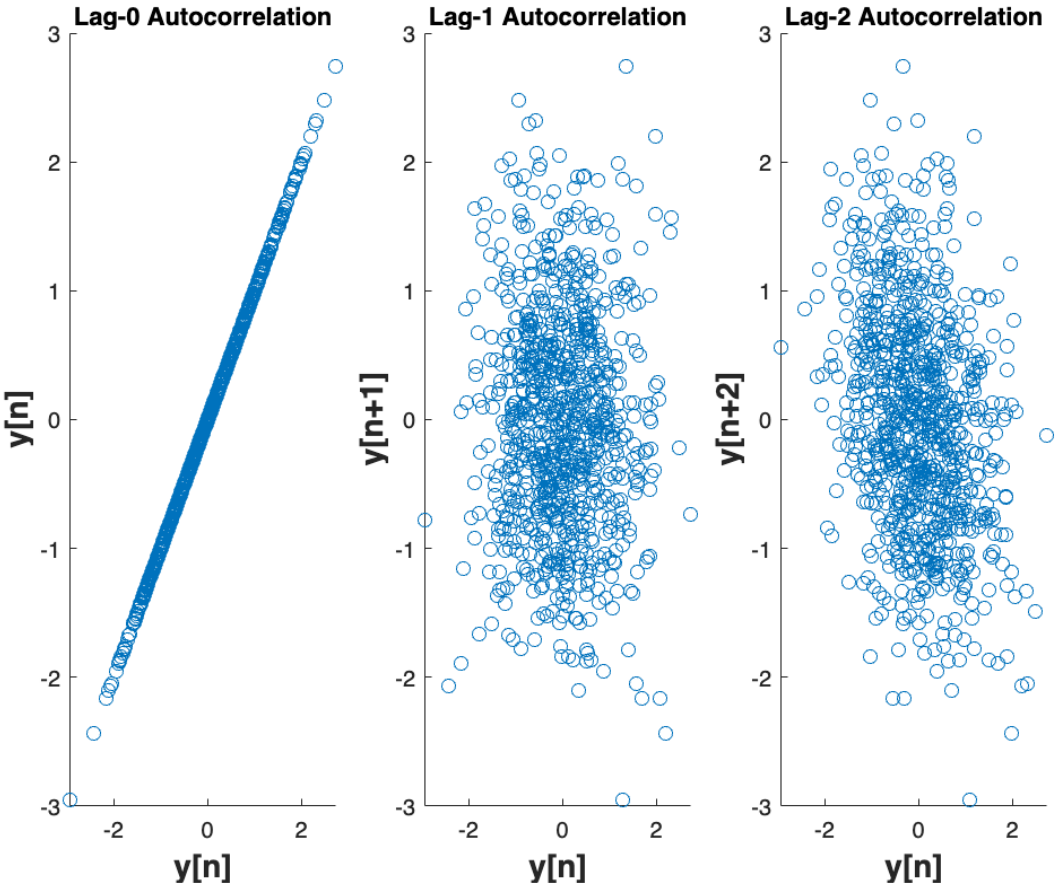
$P_y(e^{j\theta}) = \frac{3}{4} + \cos\theta + \frac{1}{2}\cos 2\theta$

c) Now, generate 1000 IID samples of $x[n]$ and use these samples to generate the samples of $y[n]$.

Generate 3 scatter plots using 'subplot(1,3,p)' (with p=1,2,3). The first scatter plot should consist of points $(y[n]; y[n])$ $(n = 1, 2, \ldots, 900)$. Notice that this correlates samples that are separated by a 'lag' of 0 samples. The other 2 scatter plots should consist of the points $(y[n]; y[n+1])$, $(y[n]; y[n+2])$, (all for $n = 1, 2, \ldots, 900$).

```
clear all
close all
clc

%------------------------------------------------
% Realization and visual inspection of the plots
%------------------------------------------------
% your code goes here (~6 lines of code, excluding the plotting code)

% Autocorrelation of y[n] for a white noise input signal

x = randn(1,1000);
y = zeros(1,1000);
for n = 3:1000
    y(n) = 1/2*(x(n) + x(n-1) - x(n-2));
end

% Compute the autocorrelation function of y[n]
Ry = zeros(3, 1);
Ry(1) = mean(y.^2);
Ry(2) = mean(y(1:end-1).*y(2:end));
Ry(3) = mean(y(1:end-2).*y(3:end));
```

```
% Generate scatter plots
figure;
subplot(1, 3, 1);
scatter(y(1:900), y(1:900));
xlabel('y[n]', 'FontSize', 14, 'FontWeight','Bold');
ylabel('y[n]', 'FontSize', 14, 'FontWeight','Bold');
title('Lag-0 Autocorrelation');

subplot(1, 3, 2);
scatter(y(1:899), y(2:900));
xlabel('y[n]', 'FontSize', 14, 'FontWeight','Bold');
ylabel('y[n+1]', 'FontSize', 14, 'FontWeight','Bold');
title('Lag-1 Autocorrelation');

subplot(1, 3, 3);
scatter(y(1:898), y(3:900));
xlabel('y[n]', 'FontSize', 14, 'FontWeight','Bold');
ylabel('y[n+2]', 'FontSize', 14, 'FontWeight','Bold');
title('Lag-2 Autocorrelation');
```



d) What can you deduce about the random process $y[n]$ from these plots?

ANSWER:

when lag=0, the $y[n]$ and $y[n]$ have a strong correlation. when lag=1, the $y[n]$ and $y[n+1]$ have weak correlation. And when lag=2, the $y[n]$ and $y[n+2]$ have nearly no correlation.

Assignment 3 - Empirical correlation and PSD function

In most real applications the following expression to estimate the autocorrelation is used:

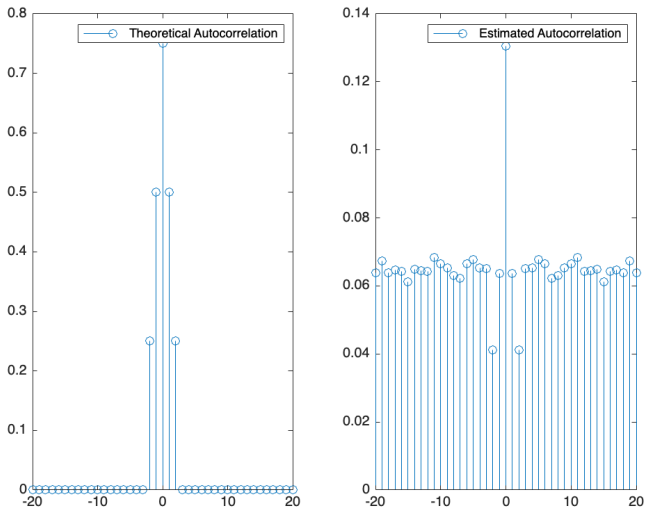
$$\hat{r_y}[l] = \frac{1}{M}\sum_{n=0}^{M-|l|-1} y[n]y[n+|l|] \quad \text{for } -(L-1) \leq l \leq (L-1)$$

where M is the number of used samples of the sequence $y[n]$ and $L \ll M$. In the next assignment we take $L = 21$.

a) Write the code to calculate $\hat{r_y}[l]$ of the LPF defined in the previous assignment. Make one stem plot with the theoretical values $r_y[l]$ (use the result from assignment 2) and the estimated values $\hat{r_y}[l]$ versus l for $-20 \leq l \leq 20$.

```
clear all
close all
clc

%-------------------------------------------------
% Evaluation of the empirical autocorrelation
%-------------------------------------------------
```

```
% your code goes here (~20 lines of code, excluding the plotting code)

% LPF coefficients from Assignment 2
M = 1000;
x = rand(1,1000);
y = zeros(1,1000);
for n = 3:1000
    y(n) = 1/2*(x(n) + x(n-1) - x(n-2));
end
y(1) = x(1);
y(2) = 0.5*(x(2)+x(1));

% Calculate the estimated autocorrelation using M samples
L = 21;
rhat_y = zeros(1, 2*L-1);
for l = -L+1:L-1
    temp_sum = sum(y(1:M-abs(l)) .* y(1+abs(l):M));
    rhat_y(l+L) = temp_sum / (M-abs(l));
end

% Calculate the theoretical autocorrelation using the coefficients
r_y = zeros(1, 2*L-1);
for i = -L+1:L-1
    r_y(i+L) = (1/4)*(3 * (i==0) + 2*(i-1==0) + (i-2==0) + 2*(i+1==0) + (i+2==0));
end

% Plot the results
figure;
subplot(1,2,1);
stem(-L+1:L-1, r_y);
legend('Theoretical Autocorrelation');

subplot(1,2,2);
stem(-L+1:L-1, rhat_y);
legend('Estimated Autocorrelation');
```




```
mean(r_y)
```

```
ans = 0.0549
```

```
mean(rhat_y)
```

```
ans = 0.0653
```

(b) For what value of lag l does $r_y[l]$ reach its maximum and for what value of lag l does $\hat{r}_y[l]$ reach its maximum?

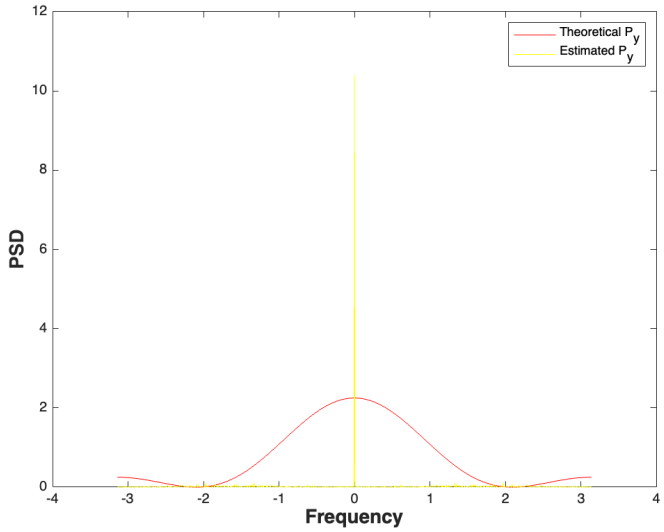
ANSWER:

Both of them reach its maximum at 0.

(c) Think of a procedure to obtain an estimate of the PSD that can be derived from the calculated values $\hat{r}_y[l]$. Apply this procedure to calculate an estimate of the PSD $\hat{P}_y(e^{j\theta})$ for θ in the range $-\pi < \theta \leq \pi$. Finally make one plot with the theoretical (use the result from assignment 2) and estimated values of the PSD.

```
%-------------------------------------------------
% Evaluation of the empirical PSD
```

```
%------------------------------------------------
% your code goes here (~3 lines of code, excluding plotting code)

% Estimate the PSD using periodogram
[psd,f] = periodogram(y,[],'centered');

% Calculate theoretical PSD using result from previous assignment
theta = linspace(-pi+1e-4,pi,1000);
the_py = 3/4 + cos(theta) + 0.5*cos(2*theta);

% Plot the theoretical and estimated PSD
figure;
plot(theta, the_py, 'r', f, psd, 'y');
xlabel('Frequency', 'FontSize',14, 'FontWeight','Bold');
ylabel('PSD', 'FontSize', 14,'FontWeight','Bold');
legend('Theoretical P_y', 'Estimated P_y');
```



(d) Give a short reasoning of possible differences between the theoretical and estimated values of the PSD.

ANSWER:

The estimation may take the errors into the calculation and caused by the number of samples. Compared to Theortical PSD, the estimated PSD have a higer variance because it is derived by finite samples.

Assignment 4 - Coss-correlation of two random processes

Cross-correlation of signals is often used in applications where we need to estimate the distance to a target (e.g., in sonar and radar).

In a basic set-up, a zero-mean signal $x[n]$ is transmitted, which then reflects off a target after traveling for $\frac{\tau}{2}$ seconds. The reflected signal is received, amplified, and then digitized to form $y[n]$.If we summarize the attenuation and amplification of the received signal by the constant α, then we can model the received signal as follows:

$$y[n] = \alpha x[n-\tau] + w[n]$$

where $w[n]$ represents additive noise from the environment and receiver electronics.

Furthermore, we assume that $x[n]$ and $w[n]$ are uncorrelated zero-mean random variables. In order to compute the distance we need to compute the delay τ. We can do this by using the cross-correlation $r_{xy}[l] = E\{x[n]y[n+l]\}$.

a) Give a short derivation in which you show the following relation: $r_{xy}[l] = \alpha r_x[l-\tau]$

ANSWER:

$r_{xy}[l] = E\{x[n]y[n+l]\} = E\{x[n](\alpha x[n+l-\tau] + w[n+l])\} = \alpha E\{x[n]x[n+l-\tau]\} + E\{x[n]w[n+l]\} = \alpha r_x[l-\tau] + 0 = \alpha r_x[l-\tau]$

b) From this result, describe shortly a procedure to estimate the delay τ based on measurements of the cross-correlation.

ANSWER:

We need to find the value of l that maximizes the cross-correlation $r_{xy}[l]$. After that, we can use the relation derived earlier: $r_{xy}[l_{max}] = \alpha r_x[l_{max} - \tau]$ to solve for the delay τ.

Assignment 5 - Estimate delay for radar data

This assignment illustrates how the cross-correlation can be used to estimate the time-delay.

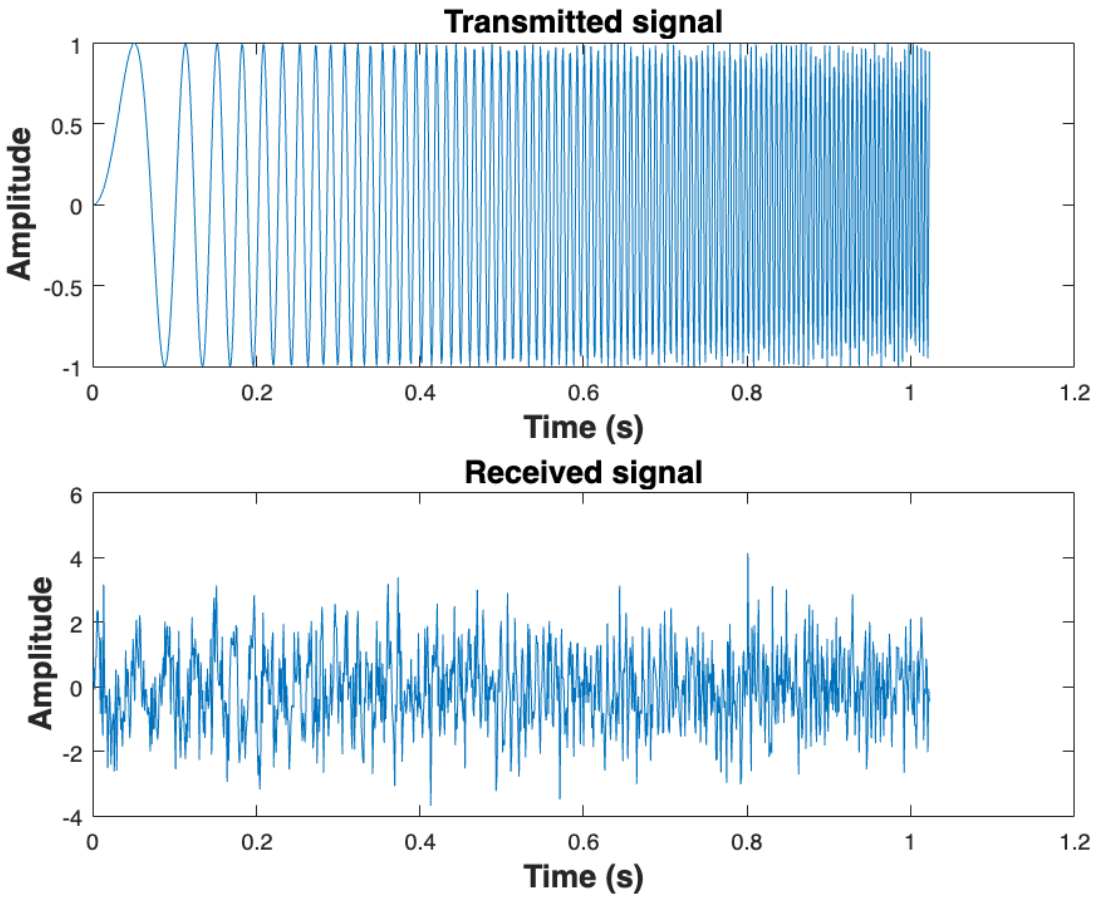
Download the MAT file radar_XX.mat (where XX represents your group number) and load it using the 'load' command. The vectors *x_tx* and *y_rx* contain two signals corresponding to the transmitted and received signals, repsectively, for a radar system.

a) Plot the transmitted and the received signals on a single figure using *subplot*. Can you estimate the delay τ by visual inspection?

ANSWER:

It is not possible to estimate the delay by visual inspection because the delay corresponds to a very small time difference between the transmitted and received signals.

```
clear all
close all
clc


%------------------------------------------------
% Plot of the data
%------------------------------------------------
% your code goes here (basically only plotting code)
% Load the data from the MAT file
load('Radar_6.mat'); % Replace XX with your group number

% Create a time vector
fs = 1e3; % Sampling frequency (Hz)
t = (0:length(x_tx)-1)/fs; % Time vector (s)

% Plot the transmitted and received signals on a single figure
figure;
subplot(2,1,1);
plot(t, x_tx);
xlabel('Time (s)','FontSize', 14, 'FontWeight','Bold');
ylabel('Amplitude','FontSize', 14, 'FontWeight','Bold');
title('Transmitted signal','FontSize', 14, 'FontWeight','Bold');
subplot(2,1,2);
plot(t, y_rx);
xlabel('Time (s)','FontSize', 14, 'FontWeight','Bold');
ylabel('Amplitude','FontSize', 14, 'FontWeight','Bold');
title('Received signal','FontSize', 14, 'FontWeight','Bold');
```



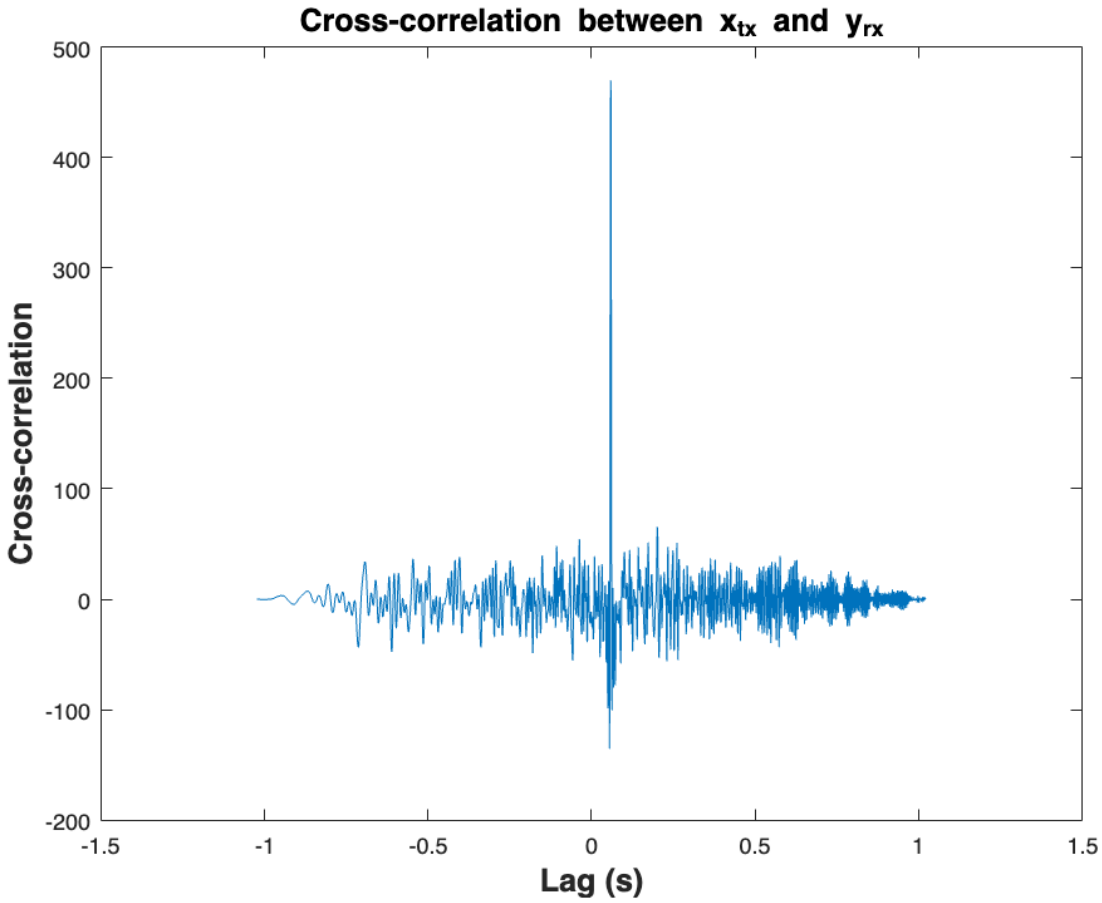
b) Use the function *xcorr* to calculate the sample cross-correlation between the transmitted and received signals. Plot the obtained cross-correlation in the range $-100 \leq l \leq 100$. Can you visually estimate the delay?

ANSWER:

It is difficult to visually estimate the delay from this plot since there are multiple peaks and the delay is very small compared to the time scale of the signals.

```
%------------------------------------------------
% Plot of the cross-correlation
```

```
%------------------------------------------------
% your code goes here (~3 lines of code, excluding plotting code)
% Calculate the cross-correlation between the transmitted and received signals
r_xy = xcorr(x_tx, y_rx);

% Create a lag vector
lags = (-length(x_tx)+1:length(x_tx)-1)/fs; % Lag vector (s)

% Plot the cross-correlation in the range -100 <= l <= 100
range = find(lags >= -100 & lags <= 100);
figure;
plot(lags(range), r_xy(range));
xlabel('Lag (s)','FontSize', 14, 'FontWeight','Bold');
ylabel('Cross-correlation','FontSize', 14, 'FontWeight','Bold');
title('Cross-correlation between x_{tx} and y_{rx}','FontSize', 14, 'FontWeight','Bold');
```



c) Implement the MATLAB function *tau=delayestimator(x,y)* which calculates the delay between two signals using the cross-correlation function.

Save the function in the same folder of this livescript file and copy the code in the section below, but not in a code section.

%----------

%Delay estimator function

%----------

% your code goes here (~8 lines of code)

```
function tau = delayestimator(x,y)
    l = length(x);
    r = xcorr(x,y,l);
    [r_max tau] = max(r);
    tau = tau - l - 1;
end
```

d) Test your function by generating two 1000-long sequences of zero-mean Gaussian random variables, denoted as $x[n]$ and $z[n]$. Then compute a new sequence $y[n] = x[n] + z[n]$. What is the estimated delay between $x[n]$ and $y[n]$?

ANSWER:

0

```
%------------------------------------------------
```

```
% Test of the function
%------------------------------------------------
% your code goes here (~4 lines of code)
% Generate two 1000-long sequences of zero-mean Gaussian random variables
x = randn(1000, 1);
z = randn(1000, 1);

% Compute the new sequence y = x + z
y = x + z;

% Use the delayestimator function to estimate the delay between x and y
tau = delayestimator(x, y);

disp(['Estimated delay between x and y: ', num2str(tau)]);
```

```
Estimated delay between x and y: 0
```

e) Use your function to calculate the delay between the transmitted and received signals.

```
%------------------------------------------------
% Evaluation of tau
%------------------------------------------------
% your code goes here (1 line of code)
tau_e = delayestimator(x_tx, y_rx);
disp(['Estimated delay between transmitted and received signals: ', num2str(tau_e)]);
```

```
Estimated delay between transmitted and received signals: 59
```

Part 2 - Estimation theory

Introduction

In this lab, we will apply concepts of estimation theory and hypothesis testing, using as a case example the diagnostic imaging for prostate cancer.

One of the most common imaging modalities used in the diagnostic workup of prostate cancer is contrast-enhanced ultrasound (CEUS). As illustrated in Fig. 1, during a CEUS examination, the patient is injected with a contrast agent and the organ of interest is scanned by using contrast-specific ultrasound scanning sequences, obtaining an ultrasound video (2D image + time). The contrast agents used for ultrasound imaging are made of microbubbles of about 5 μm in diameter. Because of their size, they cannot cross the vascular wall, and thus they are defined as intravascular contrast agents. As shown in Fig. 1, if we extract the ultrasound intensity at one single pixel over time, we obtain a time-intensity curve (TIC). When the TIC is properly linearized (inverting log-compression and time-gain compensation applied by the ultrasound scanner), then the TIC can be directly related to the changes in contrast agent concentration over time.

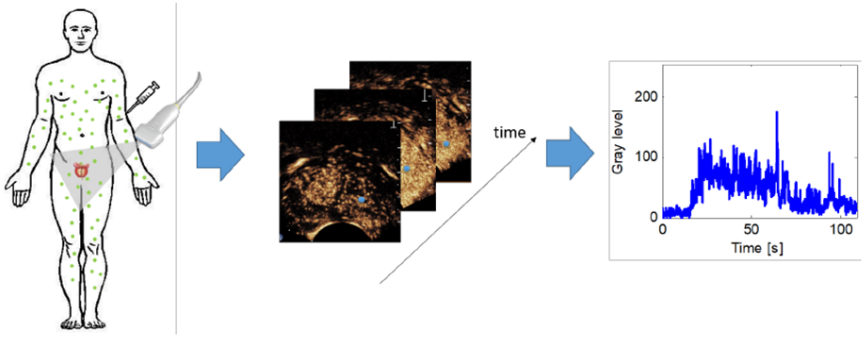


Figure 1: During a CEUS exams, the patient is injected with a contrast agent and an ultrasound video is recorded. Time-intensity curves can be extracted at each pixel by looking at ultrasound intensity changes over time.

Over the years, several models have been developed to describe the transport of contrast particles in the vasculature. In this lab, we will use the modified local density randon walk model (mLDRW). The random walk family of models for contrast concentration curves are obtained by modeling the physical process of contrast agent transport as a dispersion process superimposed to a linear drift. At any time, the contrast concentration in space is given by a Gaussian distribution. In time, the mean of this distribution moves with the same velocity as the carrier fluid (e.g., blood) and the variance increases linearly with time. This results in a skewed distribution in time. Assuming a linear relationship between the contrast agent concentration and the intensity in the ultrasound video, the mLDRW model describes the video intensity at each pixel as

$$I(t) = \mathrm{AUC}\sqrt{\frac{\kappa}{2\pi t}}e^{-\frac{\kappa(t-\mu)^2}{2t}} + I_0 \quad (1)$$

where μ represents the mean transit time, κ is a skewness parameter related to the dispersion of the contrast agent particles through the circulation, I_0 is the baseline, and the AUC is the area under the TIC.

By normalizing by the AUC and subtracting the baseline, the model in (1) can be interpreted as the probability distribution of transit time of contrast particles as

$$p(t|\Theta) = \sqrt{\frac{\kappa}{2\pi t}}e^{-\frac{\kappa(t-\mu)^2}{2t}} \quad (2)$$

where $\Theta = [\mu, \kappa]$. Normalization by the AUC in fact ensures that (2) integrates to 1.

Research has shown that the parameters of the mLDRW model are related to vascular features of tissue, which may be altered in cancer. By estimating the parameters of the mLDRW model at each pixel in the ultrasound video, a parametric map can be obtained and used to diagnose cancer. An example is shown in Fig. 2. Parameter estimation is performed by fitting TICs extracted at each pixel of the US video by the mLDRW model, thus obtaining estimated values of and at each pixel. In Fig. 2, the estimated parameter is shown in the form a colormap, which assigns a different color at each pixel depending on the parameter value. The ground truth is represented by the microscopic analysis of the prostate performed after surgical resection (radical prostatectomy). The pathologist performing the analysis marks in red the regions where cancer was present. Comparing the parametric map of κ with the ground truth, we see a good match. It suggests that the parameter κ is higher where cancer is present.

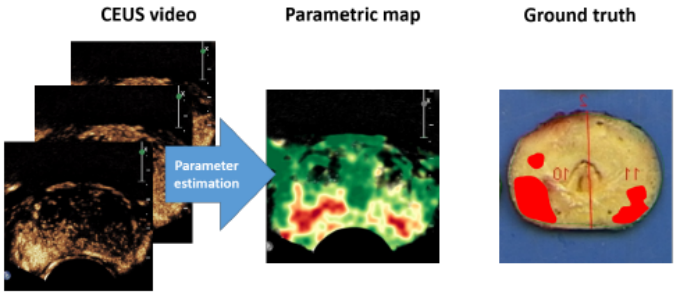


Figure 2: Example of cancer diagnosis by parametric ultrasound imaging.

1. Parameter estimation

In the following assignments, you will implement least-square estimation (LSE) and maximum likelihood estimation (MLE) to estimate the parameters in (1). The parameter AUC is given, while μ and κ are unknown. The files required to carry out the assignments are provided in the ***Lab1Part2Data*** folder.. For assignments 1 and 2, you will work with the file ***curve_fitting.mat***. This contains a vector TIC, which represents the intensity of the ultrasound video over time, a vector time, representing the time in seconds, and variable AUC, representing the given value for the parameter AUC.

The goal of the first assignment is to obtain least-square estimates for the parameters μ and κ, assuming that the model generating the observed data is given by (1). Since (1) is non-linear, you will implement non-linear LSE. For this, a template function named ***nl_lse_mldrw*** is provided in the parent folder 'Lab2data'.

Assignment 6 - Least-square estimaiton

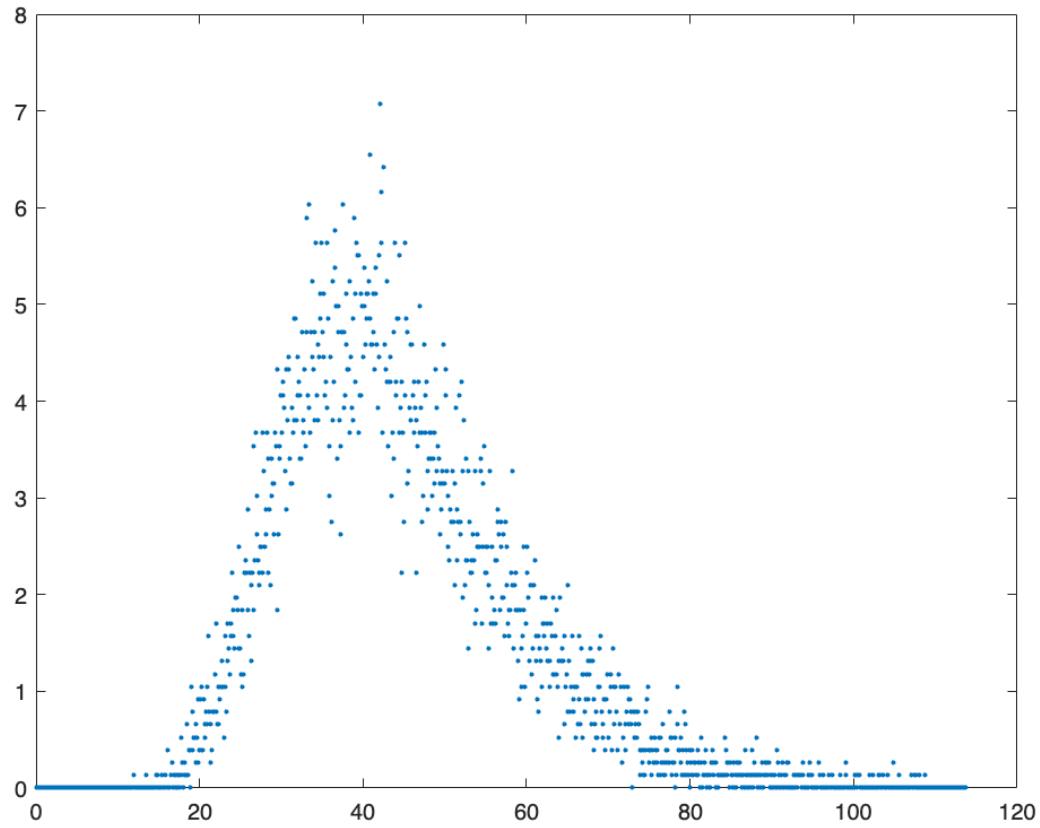
a) Derive the first derivatives of the model in (1) with respect to μ and κ, considering AUC as constant.

b) Implement a MATLAB function for non-liner LSE by the Gauss-Newton method. To do so, use the provided template function ***nl_lse_mldrw(curve, time, init_guess, maxiter, tolerance)***. Use as initial guesses for the parameter estimates $\mu_{\text{init}} = 10$ and $\kappa_{\text{init}} = 0.5$. Normalize the TIC by the AUC value and apply the implemented function to estimate the model parameters from TIC_norm=TIC/AUC (e.g., the observed TIC after the normalization).

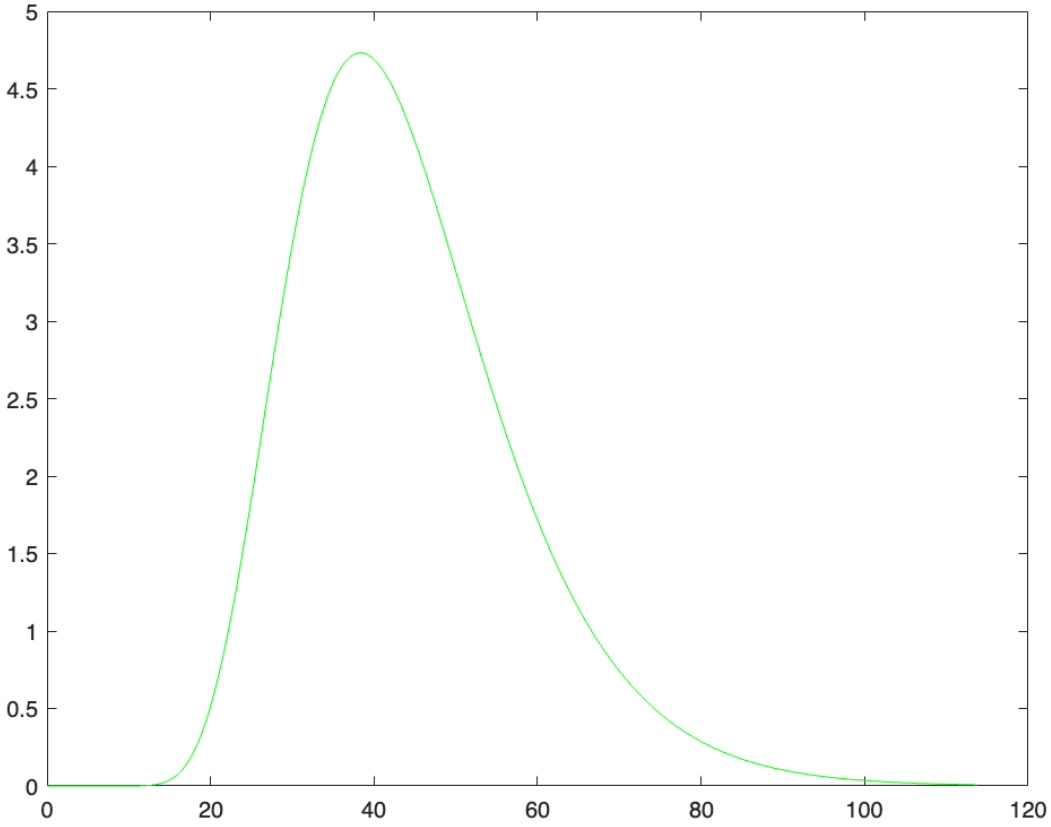
(c) Repeat curve fitting by using the built-in MATLAB function ***lsqcurvefit*** and compare the results.

Hint: Read the MATLAB documentation for lsqcurvefit. Use the appropriate option to set boundaries for the parameters estimates in order to impose a non-negative value for both μ and κ.

```
% Assignment 6 code
load("curve_fitting.mat")
% b) Complete the missing part in nl_lse_mldrw then implement LSE
plot(time',TIC,'.');
```




```
[mu, kappa] = nl_lse_mldrw(TIC/AUC,time',[10,0.5],100000,1e-16);
plot(time, AUC*sqrt(kappa./(2*pi.*time)).*exp(-kappa.*(time-mu).^2./(2.*time)), 'g');
```




```
% c) Use lsqcurvefit
```

```
fun = @(x,xdata)sqrt(x(2)./(2*pi.*xdata)).*exp(-x(2).*(xdata-x(1)).^2./(2.*xdata));
x0 = [10,0.5];
x = lsqcurvefit(fun,x0,time',TIC/AUC);
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
```

Each dataset is the result of a realization of the experiment, which is, in this example, the injection of contrast particles into the body for imaging an organ. The random variable is the arrival time of the particles at the detection point (e.g., one imaging pixel inside the organ). Points constituting the observation are drawn from a distribution. The maximum likelihood estimation (MLE) is based on the dependence between the probability of the observation of an outcome and the parameters controlling the distribution. The MLE is the answer to the question: “What should be the controlling parameters such that the observation we made is the most likely one?". To answer this question, we need the log-likelihood function $\ln \mathscr{L}$, which is the natural logarithm of the likelihood function. The Likelihood function itself is derived from the probability density function $p(t(k), \theta)$ for the samples of the observation we have, which is a function of both the data and the parameters controlling the probability density function. MLE consists of finding the controlling parameters θ that maximize the log-likelihood function, that is $\mathscr{L}(\theta) : \theta_{mle} = arg_\theta \max \ln \mathscr{L}(\theta)$.

Assignment 7 - Maximum likelihood estimaiton

(a) Derive the log-likelihood for the model in (2), at the observed times $t(k)$, with $k = 0, \ldots, K-1$.

Hint: The probability density function in (2) is continuous, but the data provided consists of measurements of number of particles at discrete time intervals. Each time sample can be seen as a single observation at $t(k) = k\Delta t$, with $k = 0, \ldots, K-1$.

(b) Derive by hand the ML estimators for μ and κ, i.e., $\widehat{\kappa}_{\mathrm{MLE}}$ and $\widehat{\mu}_{\mathrm{MLE}}$ Apply your derived estimators to estimate μ and κ. To do so, use the normalized data (TIC_norm=TIC/AUC) as in assignment 6.

(c) Repeat parameter estimation using the MATLAB command **mle**. Is the result the same as in (b)?

ANSWER:

Hint: Read the MATLAB documentation for mle. Look at the option ‘logpdf’ for using a custom log probability density function. Use the appropriate option to set boundaries for the parameters estimates in order to impose a non-negative value for both μ and κ.

```
% Assignment 7 code

% b) Implement the log-likappaelihood derived in Question a
% Estimate mu
mu_hat = sum(TIC'/AUC)/sum((TIC'/AUC)./time);
% Estimate kappa
kappa_hat = 1/(sum((TIC'./AUC).*time)/sum(TIC'/AUC) - mu_hat);

% c) Using mle from MATLAB
logpdf = @(data, mu, kappa) 1/2*sum(data.*log((kappa/2)/pi./time'))- kappa*sum((time' - mu).^2.*data./(2*time'));
para = mle(TIC./AUC,'logpdf',logpdf,'Start',[10,0.5], 'LowerBound' ,[0 0])
```

```
para = 1×2
   40.2891    0.2498
```

In the following assignment, we will test the performance of the two estimators implemented in assignments 1 and 2. We will use the file **perfomance.mat**, which includes a matrix TIC_reps containing 1000 observations of the same TIC, and the variables mu_true and kappa_true, representing the true parameter values.

Assignment 8 - Estimation performance

(a) Repeat the estimation in assignments 1 and 2 for all 1000 observations of the TIC. For each repetition, calculate the R^2 of the fit.

(b) Make a 2x2 subplot, showing the histogram of the estimated parameters, together with their true values, for the two estimators. What can you say about bias and variance of the two estimators?

ANSWER:

(c) Calculate the mean and variance of the obtained estimates. Which of the two estimators performs better in terms of bias and variance? Which estimator provides, in average, the highest R^2?

ANSWER:

```
% Assignment 8 code
load("performance.mat")
% a) repeat the estimation in assingment 1 and 2
n_reps = 1000;
R2_values = zeros(n_reps, 1);
for i = 1:n_reps
    tic_data = TIC_reps(i, :)/AUC;
    fun = @(x,xdata)sqrt(x(2)./(2*pi.*xdata)).*exp(-x(2).*(xdata-x(1)).^2./(2.*xdata));
    x0 = [10,0.5];
    xx = lsqcurvefit(fun,x0,time',tic_data');
    m = xx(1);
    k = xx(2);
    y_hat = sqrt(k./(2*pi.*time')).*exp(-k.*(time'-m).^2./(2.*time'));
    R2 = 1 - sum((tic_data - y_hat').^2)/sum((tic_data - mean(tic_data)).^2);
```

```
    R2_values(i) = R2;
end
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>

Local minimum possible.

lsqcurvefit stopped because the final change in the sum of squares relative to
its initial value is less than the value of the function tolerance.

<stopping criteria details>

Local minimum found.
```

```
RR2_values = zeros(n_reps, 1);
for i = 1:n_reps
    tic_data = TIC_reps(i, :)/AUC;
    logpdf = @(data, mu, kappa) 1/2*sum(data.*log((kappa/2)/pi./time'))- kappa*sum((time' - mu).^2.*data./(2*time'));
    para = mle(tic_data'./AUC,'logpdf',logpdf,'Start',[10,0.5], 'LowerBound' ,[0 0]);
    m = para(1);
    k = para(2);
    y_hat = sqrt(k./(2*pi.*time')).*exp(-k.*(time'-m).^2./(2.*time'));
    RR2 = 1 - sum((tic_data - y_hat').^2)/sum((tic_data - mean(tic_data)).^2);
    RR2_values(i) = RR2;
end

% b)
% c)
lse_mean_R2 = mean(R2_values);
lse_var_R2 = var(R2_values);
mle_mean_R2 = mean(RR2_values);
mle_var_R2 = var(RR2_values);
```

2. Cancer imaging

In the following assignment, we will use the implemented estimators to visualize vascular abnormalities related to cancer in the prostate. For this assignment, we will use the file **prostate_CEUS_video.mat**, which is the parent folder 'Lab2data'. The file contains a LxWxN matrix USvideo, where LxW represents the length and width of the ultrasound image, while N represent the number of time frames. The histology image is also provided as 'Groundtruth.PNG'.

Assignment 9 - Parametric maps

(a) In the matrix USvideo, each (:, :, n) represents a US image at frame n, while each (i, j, :) represents a TIC. Set n=100, and display the US image at the 100th frame. Then, choose a pixel and plot a time-intensity curve.

Hint: Use command squeeze to turn a 1x1xN matrix into a 1xN vector.

(b) Based on assignment 3, pick the best estimator and repeat the parameter estimation procedure for each pixel in the matrix. Save the parameters μ and κ estimated at each pixel in two matrices of dimension LxW. At each pixel, calculate the R^2 of the fit and set to NaN the pixels for which the $R^2 < 0.75$. For these pixels, the fit is in fact unreliable.

(c) Use the command **imagesc** to display the obtained parametric maps for μ and κ. Compare the parametric maps with the provided histology picture, where the cancer is marked in red by a pathologist. For a better visualization of the parametric map, set the limit of the colormap to a suitable range, such as the range between the 5% and 95% percentiles. Which parameter seems most promising for cancer detection?

ANSWER:

Hint: Read the the MATLAB documentation for commands *quantile* and *imagesc*. To calculate the 5% - 95% percentile range, use command *y = quantile(x,[0.05 0.95])*, where x is a vector with all the values of the parameter to be visualized. The obtained 5% - 95% percentiles can then be passed as limits for the colormap range to the function imagesc.

Note: An easy way to turn a matrix X into a vector is by typing X(:).

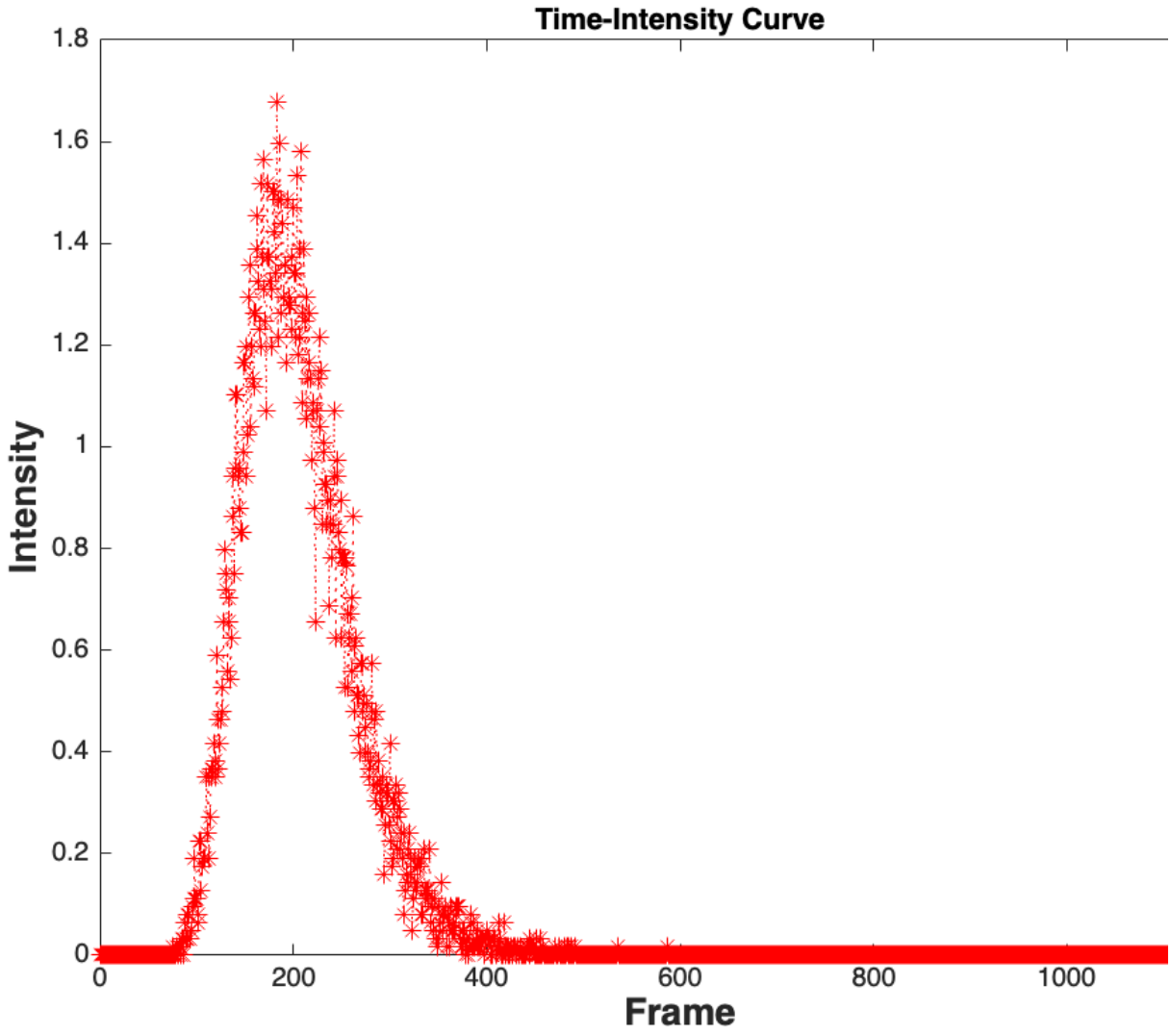
```
% Assignment 9 code

% a) displace the 100th US frame and plot a time-intensity curve
% Load the USvideo matrix (assuming it is saved in a .mat file)
load('prostate_CEUS_video.mat');

% Set n=100 to display the US image at frame 100
frame = USvideo(:,:,100);

% Choose a pixel (e.g., (50,50)) to plot a time-intensity curve
pixel_row = 50;
pixel_col = 50;
tic = squeeze(USvideo(pixel_row, pixel_col, :));

% Display the US image
figure;
```

```
imshow(frame);
```

Warning: Graphics timeout occurred. To share details of this issue with MathWorks technical support, please include that this is an unresponsive graphics client with your service request.

```
title('US Image at Frame 100');
```


```
% Plot the time-int
% ensity curve
figure;
plot(tic,'LineStyle',':','Color','r','Marker','*');
title('Time-Intensity Curve');
xlabel('Frame', 'FontSize', 14, 'FontWeight','Bold');
ylabel('Intensity', 'FontSize', 14, 'FontWeight','Bold');
```




```
% b) use the best estimator in assignment 3 to calculate unknown parameters
%mle is the best estimator.
culogpdf = @(data, mu, kappa) ...
    1/2*sum(data.*log(kappa/2/pi./time'))- kappa*sum((time' - mu).^2.*data./(2*time'));

row = size(USvideo, 1);
col = size(USvideo, 2);
mu = NaN(row,col);
kappa = NaN(row,col);

for L=1 : row
    for W=1 : col
        if isnan(AUCmap(L,W))
            continue;
        end
        frame = squeeze(USvideo(L,W,:));
        result = mle(frame./AUCmap(L,W),'logpdf',culogpdf,'Start',[10,0.5], 'LowerBound' ,[0 0]);
        mu(L,W) = result(1);
        kappa(L,W) = result(2);

        TIC_abs = squeeze(USvideo(L,W,:))./AUCmap(L,W);
        frame = sqrt(kappa(L,W)./(2*pi*(time))).*exp(-kappa(L,W)*(time-mu(L,W)).^2./(2*(time)));
        TIC_est(L,W,:) = frame;
        Rsq1(L,W) = 1 - sum((TIC_abs - frame').^2)/sum((frame' - mean(frame').^2));
        if Rsq1(L,W) < 0.75
            Rsq1(L,W) = NaN;
        end
    end
end

save q9.mat mu kappa Rsq1
close all
imshow(Rsq1)
```

```
% c) visulize parametric maps
load q9.mat
mu_clims = quantile(mu(:), [0.05 0.95]);
kappa_clims = quantile(kappa(:),[0.05 0.95]);
subplot(1,2,1)
imshow(mu, mu_clims)
subplot(1,2,2)
imshow(kappa, kappa_clims)
```