Lab 1: Random variables and estimation theory

Statistical Signal Processing (5CTA0) - Lab Assignment

Lecturers: Simona Turco (Flux 7.076), Franz Lampel (Flux 7.064)

Assistants: Yizhou Huang, Ben Luijten, Agata Barbagini, Xueting Li

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 4 students, which must register 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, running 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 emailing the lab assistants. The email address are available on Canvas.
- 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 functions (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 commandsfor 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 10/10/22 at 23:59.

Introduction

In this lab, we are going to prove the frequentist definition of probability and investigate sampling of random variables and the central-limit theorem in MATLAB. We are also going to implement maximum-likelihood estimator for noisy ultrasound signals.

Assignment 1 - Probability and frequency

In the first assignment, we are going to explore a traditional probability problem: rolling a dice.

The *classic* definition of probability states that if a random experiment can result in N mutually exclusive and equally likely outcomes, and if an event A results from the occurrence of N_A outcomes, then the probability A is defined as

$$Pr[A] = \frac{N_A}{N}. \qquad (1)$$

The *frequentist* definition of probability, also known as *relative probability*, calculates probability based on how often an event occurs. The relative frequency of an event is given by

$$f_A = \frac{\text{number of occurrences of event } A}{\text{total number of observations}} = \frac{N(A)}{A} \qquad (2)$$

where $N(\cdot)$ denotes the number of occurrences of a certain event and N the total number of observations. The relative frequency can be understood as how often event A occurs relative to all observations.

QUESTION 1a

If you have a dice which has six faces numbering from 1 to 6. You roll the dice for six rounds.

What is the probability of getting a 1 during all rounds?

What is the probability of getting all 1s?

What is the probability of only getting 1s at the third and the fourth rounds whereas the other numbers at other rounds (like XX11XX)?

Write down your problem-solving process.

ANSWER: (write solution here. You can insert equations by going to 'Insert', and then 'Equations')

First, the probability of throwing a 1 with an unbiased die is naturally 1/6.

The probability of throwing a certain number of 1s is given by a binomial distribution, or

$P_X(x) = \binom{6}{x}(p)^x(1-p)^{6-x}$, where $p = 1/6$.

Event A: throwing a single 1 during all six rounds

$P_X(1) = \binom{6}{1}(1/6)(5/6)^5 = 0.4019.$

Event B: throwing a 1 every round

$P_X(6) = \binom{6}{0}(1/6)^6(5/6)^0 = 2.1433 \cdot 10^{-5}.$

Event C: throwing a 1 only on the third and fourth round

Note that here, the 1s are required to be in a certain position, so the binomial distribution no longer holds.

There is only one way to throw this pattern, thus the probability is given by

$P[\text{two 1s in the middle}] = (1/6)^2(5/6)^4 = 0.0134.$

QUESTION 1b

Now, you are going to implement a MATLAB script to prove that the results you obtained from question **1a** are correct.

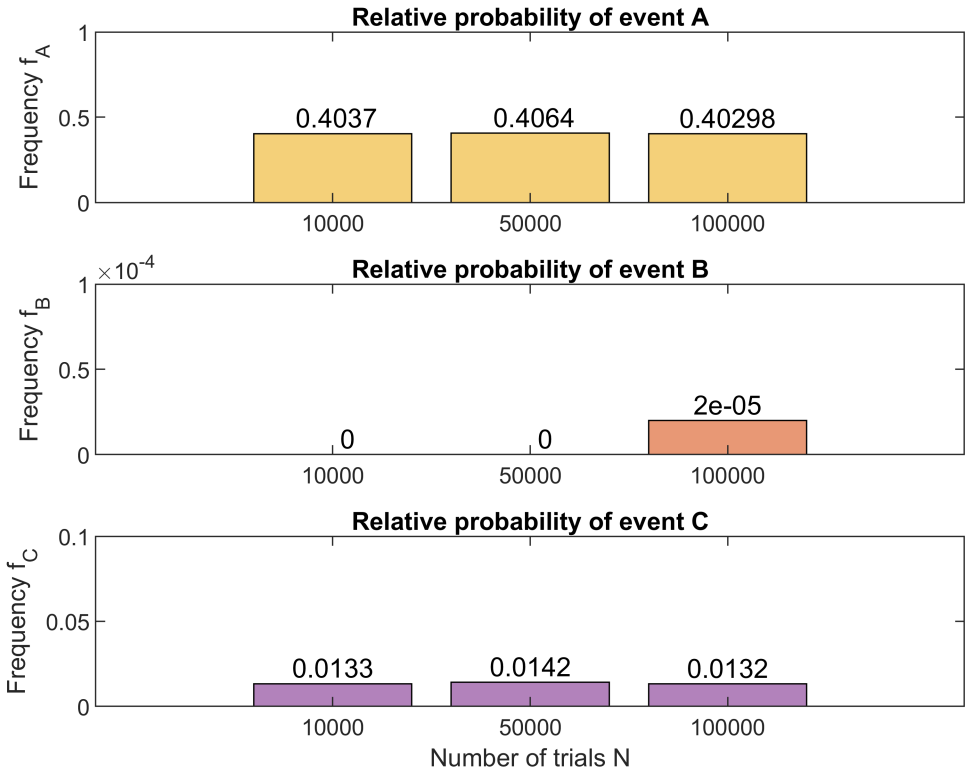
Let's take 'What is the probability of getting all 1s? ' as an example.

Please write a script to repeat the experiment for 10000, 50000 and 100000 times and calculate the relative frequency of getting all 1s if you roll the dice six times. Compare the results obtained by 10000, 50000 and 100000 times with your theoretical result in question **1a**, do you get the same result? If not, why?

HINT: A useful command to simulate this experiment is `random`. (optional) To avoid having a different result every time you run the algorithm, you may fix the random seed using the function `rng`.

```
%% Initialize the simulation
% Fix the seed
seed = 6;
rng(seed);

N = [10000 50000 100000];   % Number of trials

% Store the number of times each event happens, for each value of N
count_A = zeros(1,3);
count_B = zeros(1,3);
count_C = zeros(1,3);

%% Run the simulation
% Set the number of trials
for k = 1:3
for n = 1:N(k)

    % Reset and roll the 6 dice
    dice = zeros(1,6);
    for i = 1:6
        dice(i) = unidrnd(6);   % ~ Uniform(1,6) (discrete)
    end

    % Note; events A, B and C are all disjoint (as follows from their
    % definition, so elseif-statements can be used.
    % Check for Event A (a single 1)
    if sum(dice == 1) == 1
        count_A(k) = count_A(k) + 1;

    % Check for Event B (all 1s)
    elseif sum(dice == 1) == 6
        count_B(k) = count_B(k) + 1;

    % Check for Event C (third and fourth throws are 1s)
    elseif dice(3) == 1 && dice(4) == 1 && sum(dice == 1) == 2
        count_C(k) = count_C(k) + 1;
    end
end
end

%% Plot the resulting frequentist probabilities
figure(1)

subplot(3,1,1) % Plot event A
bar(count_A./N,'FaceColor',[0.9290 0.6940 0.1250],'FaceAlpha',0.6)
title('Relative probability of event A')
%xlabel('Number of trials N')
ylabel('Frequency f_A')
xticklabels({'10000','50000','100000'})
```

```
ylim([0 1])
text(1:length(count_A),count_A./N,num2str((count_A./N)'),...
    'vert','bottom','horiz','center'); % Plot values on top of the bars

subplot(3,1,2) % Plot event B
bar(count_B./N,'FaceColor',[0.8500 0.3250 0.0980],'FaceAlpha',0.6)
title('Relative probability of event B')
%xlabel('Number of trials N')
ylabel('Frequency f_B')
xticklabels({'10000','50000','100000'})
ylim([0 0.0001])
text(1:length(count_B),count_B./N,num2str((count_B./N)'),...
    'vert','bottom','horiz','center');

subplot(3,1,3) % Plot event C
bar(count_C./N,'FaceColor',[0.4940 0.1840 0.5560],'FaceAlpha',0.6)
title('Relative probability of event C')
xlabel('Number of trials N')
ylabel('Frequency f_C')
xticklabels({'10000','50000','100000'})
ylim([0 0.1])
text(1:length(count_C),count_C./N,num2str((count_C./N)'),...
    'vert','bottom','horiz','center');
```



ANSWER:

For event A (a single 1), the three number of trials N all provide a decent approximation of the theoretical probability of 0.4019, although the highest amount of trials (N = 100000) gives the smallest difference.

For event B (all 1s), the event occurs zero times for N = 10000 and N = 50000, which implies that the relative probability is zero. The theoretical probability, however, is not zero but simply very small (2e-5). As a result, it is probable that the simulation will not encounter this event for a low amount of trials. Thus, a sufficiently large amount of trials is required when simulating events with low probabilities.

For event C (1s on throws three and four), the experimental results again match the theoretical probability relatively well.

Assignment 2 - Moments of a random variable

In the following experiment, we construct a random variable $X_N[n]$ as the sum of N RVs x_i as follows:

$$X_N = \frac{1}{N}\sum_{i=1}^{N} x_i[n] \quad \text{with } N = 1, 2, \ldots \tag{3}$$

The random processes $x_i[n]$, for $i = 1, 2, \ldots, N$, are N IID stationary Gaussian random processes with mean $E\{x_i[n]\} = \mu_i = \mu$ and variance $E\{x_i^2[n]\} = \sigma_i^2 = \sigma^2$.

QUESTION 2a

Calculate an analytical expression for the mean $\mu_{X_N} = E\{X_N\}$ and the variance $\sigma_{X_N}^2 = E\{X_N^2\} - E\{X_N\}^2$ of the sum RV X_N as a function of N.

HINT: The x_i are IID (independent, identically distributed)

ANSWER: Please insert the analytical expression here.

Starting off, the mean of X_N can be found to be:

$$E[X_N] = E\left[\frac{1}{N}\sum_{i=1}^{N} x_i[n]\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N} E[x_i[n]]$$

$$= \frac{1}{N} N\mu_i$$

$$= \mu.$$

Next, the variance can be given by:

$$\sigma_{X_N}{}^2 = \mathrm{Var}[X_n]$$

$$= \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} x_i[n]\right]$$

$$= \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}[x_i[n]]$$

$$= \frac{1}{N^2} N\sigma_i^2$$

$$= \frac{\sigma^2}{N} .$$

QUESTION 2b

Calculate an analytical expression for the expectation of $E\{X_N^3\}$, the 3rd moment of X_N, $M_X{}^3$, and the skewness of the random process.

ANSWER: Please insert the analytical expression here.

Considering X_N is formed by summing up N IID Gaussians, X_N will also follow a Gaussian distribution.

Furthermore, consider the fact that the PDF of X_N is symmetrical. As a result, the skewness ($\widetilde{\mu}_3$) of this random process is 0 by definition.

The relationship between the skewness and the third moment is as follows:

$$\widetilde{\mu}_3 = \frac{E[(X_N - \mu_{X_N})^3]}{\sigma_{X_N}{}^3}$$

Note that the upper part of the fraction can be rewritten to

$$E[X_N{}^3] - 3\mu_{X_N} E[X_N{}^2] + 2\mu_{X_N}{}^3,$$

with $E[X_N{}^2] = \mu^2 + \frac{\sigma^2}{N}$ as a result of the definition of the variance.

For this equation to result in 0, the third moment is found to be

$$E[X_N{}^3] = \mu^3 + 3\mu\frac{\sigma^2}{N}.$$

QUESTION 2c

Implement a Monte-Carlo simulation in MATLAB to generate RVs x_i, for $i = 1, 2, \ldots, 5$, with a gaussian distribution. For each of these RVs, generate 1000 IID samples ($n = 1000$) with parameters of the exponential distribution with $\mu = 5$ and $\sigma = 2$. Construct X_{gN} for $N = 1,\ 2,\ \mathrm{and}\ 5$ and make a *subplot* of X_{g1}, X_{g2} and X_{g5}. Calculate mean, variance and skewness of X_{g5}.

```
%% Fix the seed
seed = 6;
rng(seed);

%% Set parameters
```

```
mu = 5;          % Mean
sigma = 2;       % Standard deviation
n = 1000;        % Number of samples per RV
Nmax = 5;        % Max. number of RVs

%% Generate random variables
% Each row of x corresponds to a Gaussian random variable x_i, with the
% columns containing its n samples.
x = normrnd(mu,sigma,Nmax,n);          % x ~ Normal(mu,sigma)

Xg = cumsum(x)./(1:Nmax)';
% By calculating the cumulative sum and normalizing it by N in [1,Nmax],
% the nth row of Xg now indeed holds Xgn.

%% Statistical properties for Xg5
% Simulated results
mu_sim = mean(Xg(5,:));              % Mean
sigma2_sim = var(Xg(5,:));           % Variance
skew_sim = skewness(Xg(5,:));        % Skewness

%% Plotting
edges = -1:0.4:11;

figure(1)
subplot(3,1,1), histogram(Xg(1,:),edges,'Normalization','pdf',...
    'FaceColor',[0.9290 0.6940 0.1250])
title('Histogram of samples X_{g1}')
ylabel('Frequency')
xlim([-1 11])
ylim([0 0.5])
yticks([0 0.25 0.5])

subplot(3,1,2), histogram(Xg(2,:),edges,'Normalization','pdf',...
    'FaceColor',[0.8500 0.3250 0.0980])
title('Histogram of samples X_{g2}')
ylabel('Frequency')
xlim([-1 11])
ylim([0 0.5])
yticks([0 0.25 0.5])

subplot(3,1,3), histogram(Xg(3,:),edges,'Normalization','pdf',...
    'FaceColor',[0.4940 0.1840 0.5560])
title('Histogram of samples X_{g5}')
xlabel('Sample value')
ylabel('Frequency')
xlim([-1 11])
ylim([0 0.5])
yticks([0 0.25 0.5])
```

QUESTION 2d

Compare the results obtained from question **2a** and **2b** with your implementation in question **2c**. Are they the same? If not, why? Can you make a general statement on the sum of multiple independent Gaussian distributed random processes? (for instance which distribution it follows)

ANSWER:

The overall trends are as expected; for all N considered (1, 2 and 5) the mean lies around 5, however, as N grows the variance decreases.

The analytical answers give: mean = 5, variance = 0.8, skewness = 0.

The simulated results yield: mean = 5.0473, variance = 0.8098, skewness = 0.0263.

Thus, the simulation results are in good agreement with the analytically obtained descriptors.

The sum of multiple independent Gaussian random variables follows a Gaussian distribution as well.

In probability theory, the **central limit theorem** (**CLT**) establishes that, in many situations, when independent random variables are summed up, their **normalized sum** tends toward a normal distribution even if the original variables themselves are not normally distributed.

QUESTION 2e

Please write down the analytical expression of the normalized sum of N random processes $x_i[n]$, Z_N, as a function of mean μ_{XN} standard deviation σ_{XN}.

ANSWER: Please insert the analytical expression here.

$$Z_N = \frac{1}{N}\sum_{i=1}^{N}\frac{x_i[n]-\mu}{\sigma^2} = \frac{1}{N\sigma^2}\left(\sum_{i=1}^{N}x_i[n]-\mu N\right) = \frac{1}{\sigma^2}X_n - \frac{\mu}{\sigma^2} = \frac{1}{\sigma^2}(X_N-\mu)$$

$$\sigma_{X_N}{}^2 = \frac{\sigma^2}{N}$$

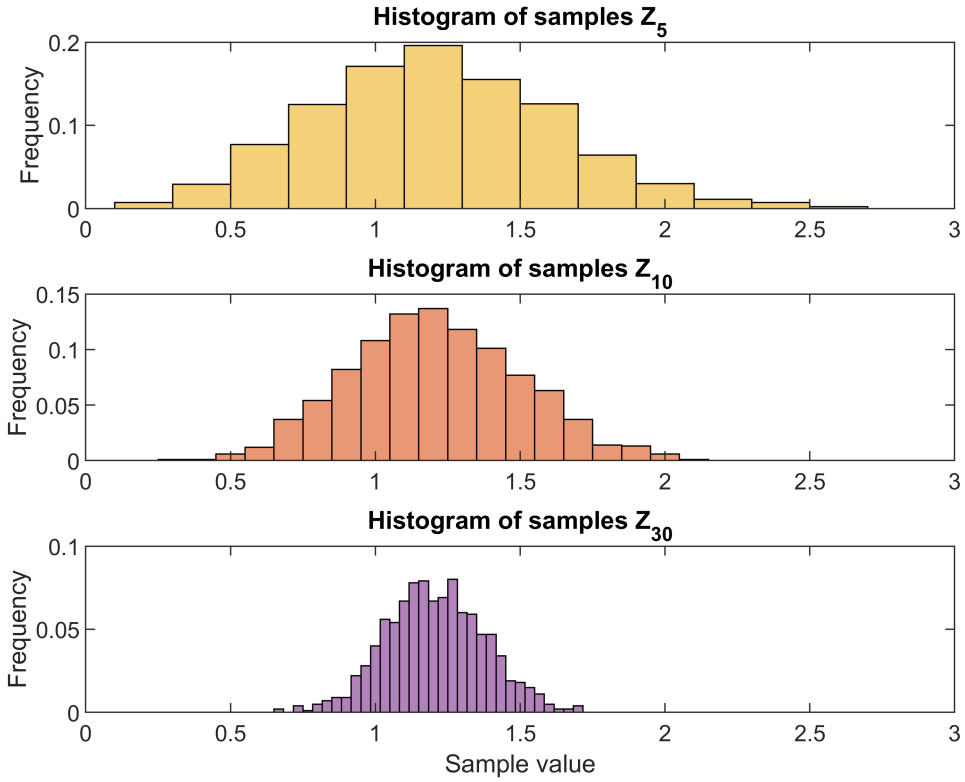
$$Z_N = \frac{X_N - \mu_{X_N}}{N\sigma_{X_N}{}^2}$$

QUESTION 2f

Repeat the implementation in question **2c**, but this time replace the gaussian distribution with a binomial distribution with $n = 6$ and $p = 0.2$. Try $N = 5, 10 \text{ and } 30$, and plot the histograms of Z_5, Z_{10}, Z_{30}. Does your implementation converge to Gaussian distribution when N is 'sufficiently' large? Please comment on this.

```
%% Fix the seed
seed = 6;
rng(seed);

%% Set parameters
```

```
% Binomial distribution
n_bin = 6;      % Number of trials
p = 0.2;        % Probability of success

n = 1000;       % Number of samples per RV
Nmax = 30;      % Max. number of RVs

%% Generate random variables
% See exercise 2c for relevant comments
x = binornd(n_bin,p,Nmax,n);
Z = cumsum(x)./(1:Nmax)';

%% Plotting
figure(1)
subplot(3,1,1), histogram(Z(5,:),(-1/10):(1/5):3,...
    'Normalization','probability','FaceColor',[0.9290 0.6940 0.1250])
title('Histogram of samples Z_{5}')
ylabel('Frequency')
xlim([0 3])
ylim([0 0.20])

subplot(3,1,2), histogram(Z(10,:),(-1/20):(1/10):2.5,...
    'Normalization','probability','FaceColor',[0.8500 0.3250 0.0980])
title('Histogram of samples Z_{10}')
ylabel('Frequency')
xlim([0 3])
ylim([0 0.15])


subplot(3,1,3), histogram(Z(30,:),(-1/60):(1/30):2,...
    'Normalization','probability','FaceColor',[0.4940 0.1840 0.5560])
title('Histogram of samples Z_{30}')
xlabel('Sample value')
ylabel('Frequency')
xlim([0 3])
ylim([0 0.10])
```

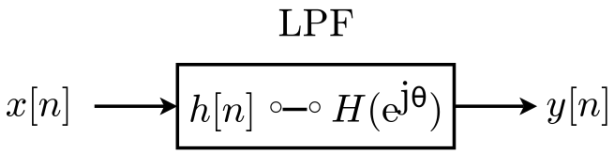


ANSWER:

By the Central Limit Theorem, Z_N should converge to a Gaussian distribution as N becomes sufficiently large. This implementation does converge to a Gaussian distribution for a sufficiently large N, however, it converges more slowly compared to the case of the sum of Gaussians in question 2c. This could be attributed to the initial shape of the PDF of a binomial distribution and its discrete nature.

Assignment 3 - Auto-correlation and Cross-correlation

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 process. We will study the statistical properties of the output process and its relation with the input process and the filtering operation. The figure below is 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 the first instance we are using the very simple LPF:

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

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

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

which implies that the output samples are averaged values of 2 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:

$$r_x[l] = E\{y[n]y[n-l]\} = E\left[\frac{1}{4}(x[n] + x[n+1])(x[n-l] + n[n-1-l])\right] =$$

$$= \frac{1}{4}(E(x[n]x[n-l]) + E(x[n]x[n-1-l]) + E(x[n-1]x[n-l]) + E(x[n-1]x[n-1-l])) =$$

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

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

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 $P_y(e^{j\theta}) = P_x(e^{j\theta}) \cdot |H(e^{j\theta})|^2$.

ANSWER:

1)

$$P_y(e^{j\theta}) = \sum_{l=-\infty}^{+\infty} r_y[l](e^{-\mathrm{j}l\theta}) = \frac{1}{4}\sum_{l=-\infty}^{+\infty}((2\delta[l] + \delta[l+1] + \delta[l-1]))e^{-\mathrm{j}l\theta} =$$

$$= \frac{1}{4}(2 + e^{j\theta} + e^{-j\theta}) = \frac{1}{4}(2 + 2\cos(\theta)) = \frac{1}{2}(1 + \cos(\theta)) = \cos^2\left(\frac{\theta}{2}\right)$$

2)

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

$$H(z) = \sum_{n=-\infty}^{+\infty} h[n]z^{-n} = \frac{1}{2}\sum_{n=-\infty}^{+\infty} \delta[n] + \delta[n-1]z^{-n} = \frac{1}{2}(1 + z^{-1)})$$

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

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

$$P_y(e^{j\theta}) = \cos^2\left(\frac{\theta}{2}\right)$$

As can be seen, the two methods result in the same PSD.

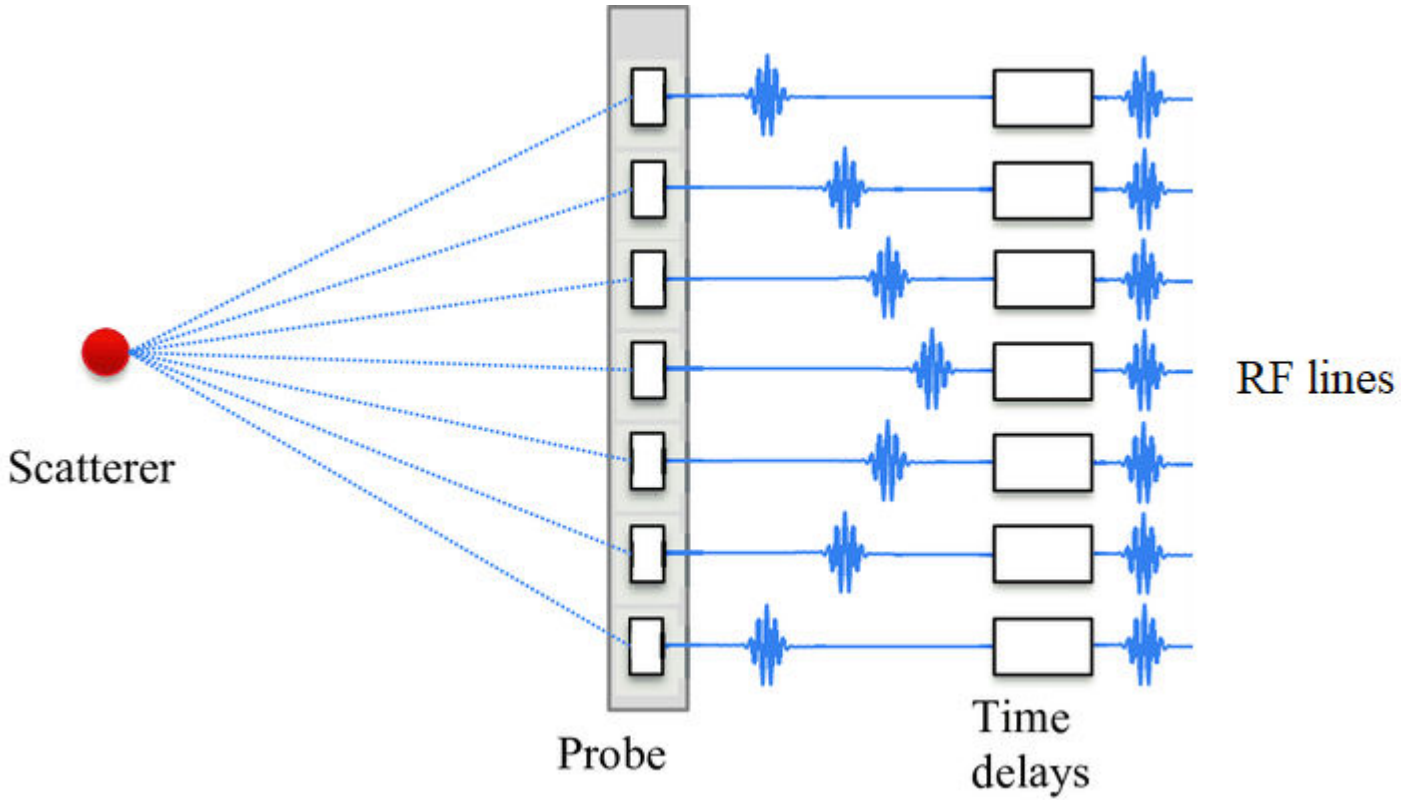
Part 2 - Estimation Theory

Ultrasound imaging is based on the pulse-echo principle. First, a pressure pulse is transmitted towards a region of interest by the ultrasound transducer consisting of multiple transducer elements (sensors). Within the medium X, scattering occurs due to inhomogeneities in density, speed-of-sound and other non-linear behavior. The resulting back-scattered echoes are recorded using the same transducer elements, yielding a set of radio-frequency (RF) signals Y with added noise N.

This can be described by the linear model

$$Y = A X + N \qquad (4)$$

where A is a measurement matrix that models the wave propagation (i.e. the travel time) and array response.



In order to create an image, we need to focus the received signals to each image pixel. This is done by applying delays to the received signal based on their respective travel time. This process is also known as time-of-flight

correction. After time-of-flight correction our model can be written in vector form for each pixel independently such that.

$$\boldsymbol{y} = \boldsymbol{a}_{\theta}\, x + \boldsymbol{n} \qquad (5)$$

where $\boldsymbol{y} \in \Re^{m}, x \in \Re^{1}$ and $n \in \Re^{m}$ denote the delayed RF signals, the scattering intensity, and additive noise, respectively, for *m* elements . The array response is denoted by $\boldsymbol{a}_{\theta} \in \Re^{m}$.

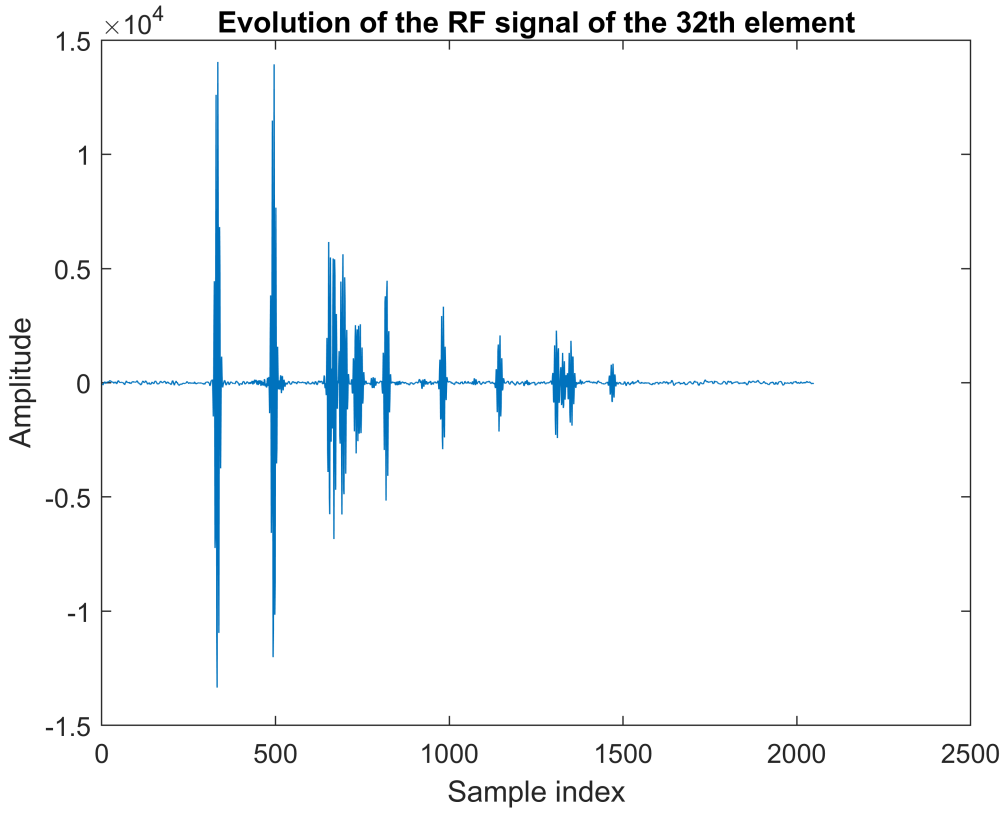
In order to reconstruct an interpretable image from our measurements, we need to find the scattering intensity. It is however not possible to find the exact value of *x,* due to noise. In this assignment we will implement a maximum likelihood estimator for reconstructing an ultrasound image from a set of noisy measurement.

Assignment 4 - Data processing and time-of-flight correction.

QUESTION 4a

Load the ultrasound data in **rf_data.mat** from a set of point-scatterers. The data comprises 64 elements, with each 2048 time samples, stored in the format (time samples, elements), Plot the RF signal of the 32th (center) element over time.

```
% Load and plot the data (~3 lines of code)
load('rf_data.mat')

figure()
plot(Y(:,32))
xlabel('Sample index')
ylabel('Amplitude')
title('Evolution of the RF signal of the 32th element')
```



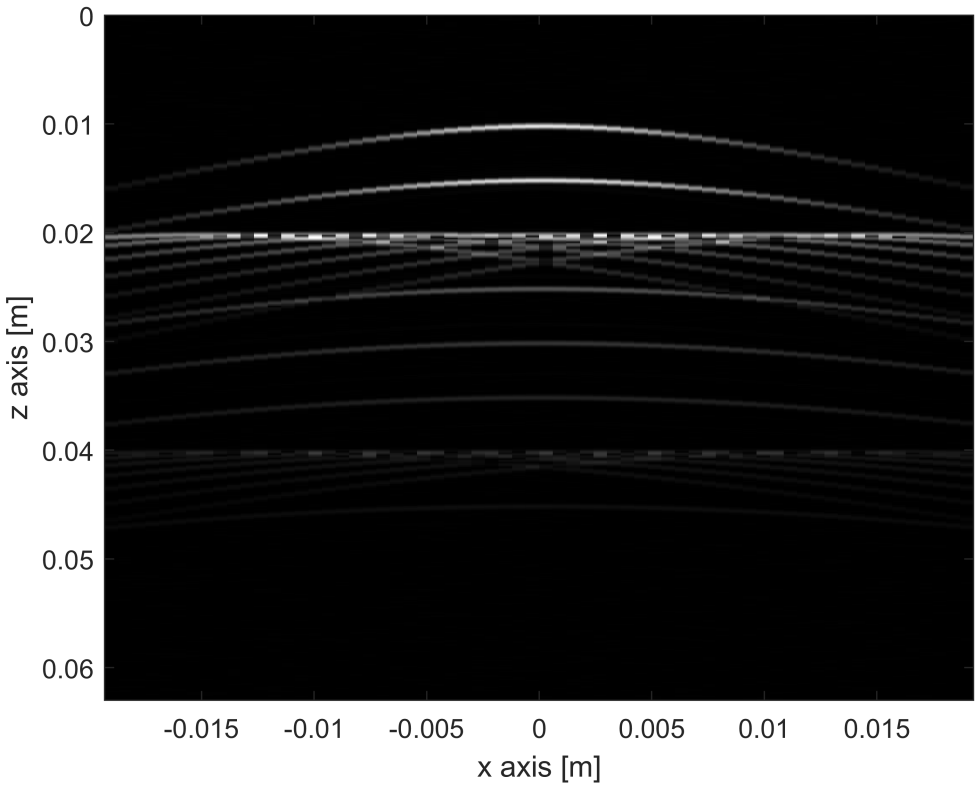
QUESTION 4b

The plot from **4a** is also called an A-mode (amplitude) image. Often, it is more informative to reconstruct a so called B-mode (brightness) image, which provides a 2D scattering map of the imaging region. Lets first take the envelope of the received signal. An easy way of finding the envelope of a modulated signal (without knowing the carrier frequency) is by taking the absolute value of the Hilbert transform `hilbert()`.

Visualize the resulting envelope image.

HINT: Use imagesc() for plotting

```
% Generate axis from our imaging region
sz = size(Y);
x_axis = linspace(-19e-3,19e-3, sz(2)); % -19mm to 19mm wide
z_axis = linspace(0,63e-3, sz(1)); % 0mm to 63mm in depth

% Envelope detection
envelope = hilbert(Y);
envelope = abs(envelope);

% Visualization
figure()
colormap gray; axis image;
imagesc(x_axis, z_axis, envelope)
xlabel('x axis [m]')
ylabel('z axis [m]')
```

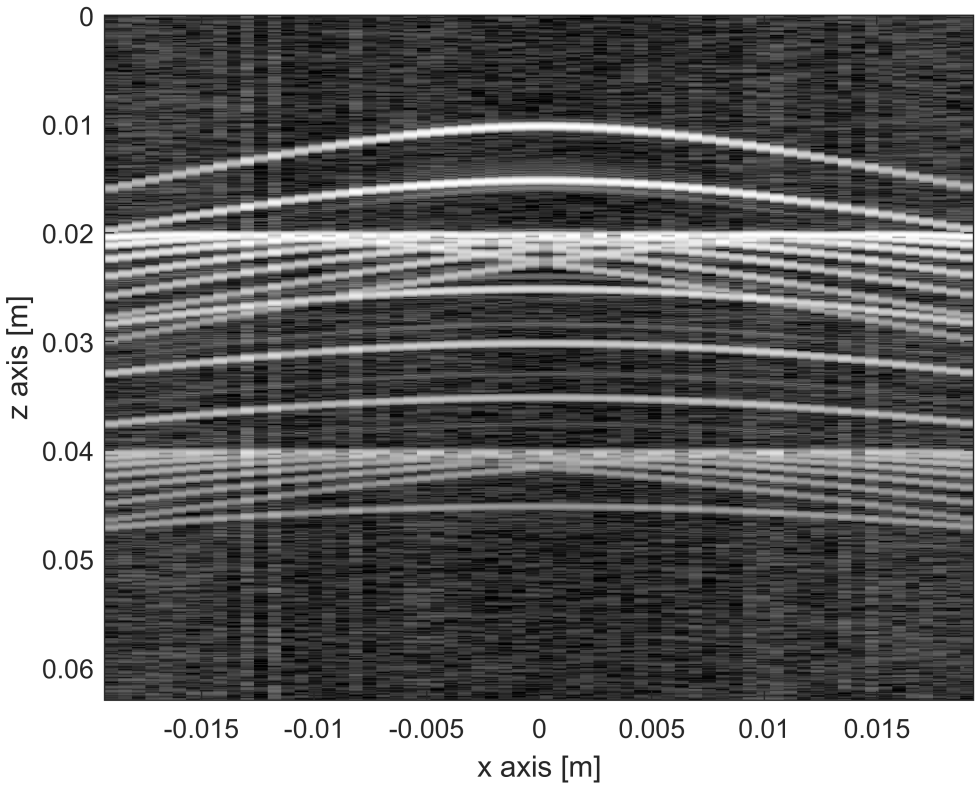


QUESTION 4c

Since ultrasound signals have a high dynamic range, the data is typically logarithmically compressed (i.e. converted to dB scale) to yield an interpretable image.

Repeat the steps from **4b**, but now normalize the envelope data such that it ranges between [0,1], and convert it to decibels. Plot the resulting image with a dynamic range of 60dB, what differences do you see?

```
% Log compression
envelopedb = 20 * log10(envelope ./ max(envelope));

% Visualization
figure;
colormap gray; axis image;
imagesc(x_axis, z_axis, envelopedb, [-60 0])
xlabel('x axis [m]')
ylabel('z axis [m]')
```



ANSWER:

It is easier to distinguish the different parts of the enveloppe in the logarithmically compressed image, because the lower values which are not visible at first are amplified ; hence more lines are visible the second plot.

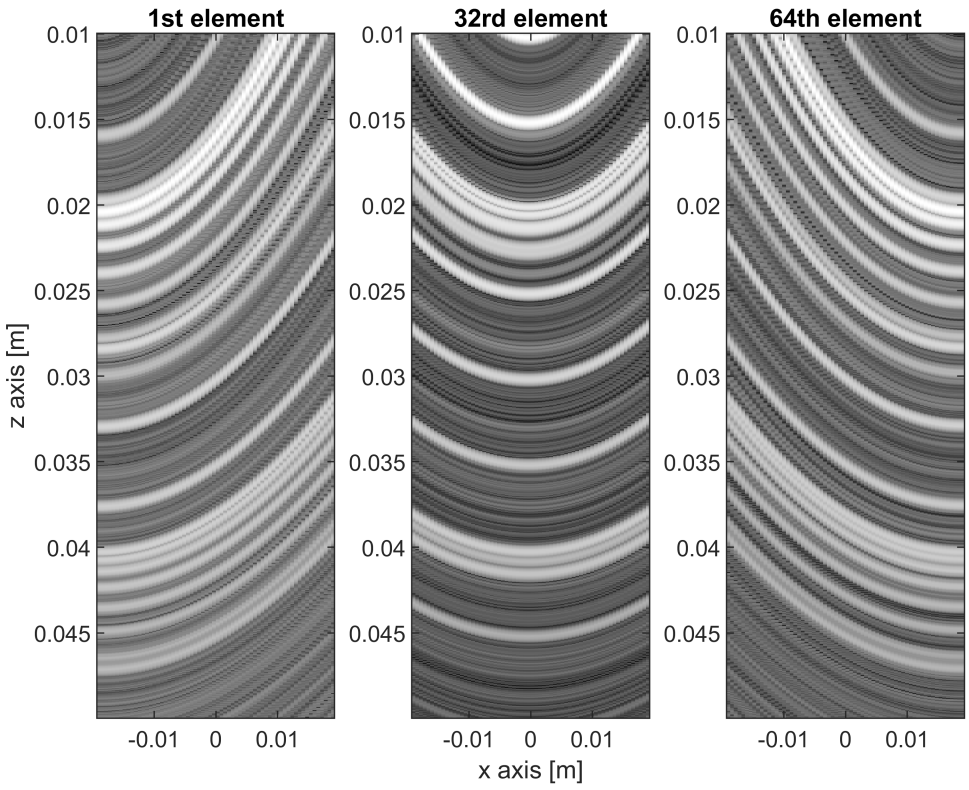
QUESTION 4d

So far we have visualized the raw unfocussed RF signals. We are now going to focus the signals using time-of-flight correction. Use the provided `tof_correction()` function to align the RF signals to each pixel. The resulting matrix has the shape `(x, z, elements)`. In other words, for each pixel in our grid, we get 64 individual signal contributions.

Apply time of flight correction and plot the images corresponding to the 1st, 32th and 64th element, what do you see?

```
% TOF grid definition
% A sub-region is choosen to speed up calculations
z_axis = z_axis(325:1625); % 10mm to 50mm depth
[x_grid, z_grid] = meshgrid(x_axis, z_axis);


% Apply time-of-flight correction and plot the result for the 3 different
```

```
% elements.
corrected_signal = tof_correction(envelopedb, x_grid, z_grid);

% Visualization
figure()

colormap gray; axis image;
subplot(1, 3, 1)
imagesc(x_axis, z_axis, corrected_signal(:,:,1))
title('1st element')
ylabel('z axis [m]')

subplot(1, 3, 2)
imagesc(x_axis, z_axis, corrected_signal(:,:,32))
xlabel('x axis [m]')
title('32rd element')

subplot(1, 3, 3)
imagesc(x_axis, z_axis, corrected_signal(:,:,64))
title('64th element')
```



ANSWER:

The focused RF signals have varying shapes depending on the element number chosen. This is due to the difference in location of the probes, thus resulting in different RF line profiles.

Assignment 5 - Maximum Likelihood estimation

QUESTION 5a

To reconstruct the underlying tissue scattering x, we need to invert the model presented in (2).

Under the assumption of white gaussian noise $\boldsymbol{n} \sim N(0, \sigma_n^2 I)$, and $\boldsymbol{a}_\theta = 1$ (uniform array response), find the Maximum-Likelihood Estimator for x.

ANSWER:

The maximum likelihood estimator can be found using the derivative of the log-likelihood function. Since $\boldsymbol{y} \sim N(\boldsymbol{a}_\theta\, x,\, \sigma_n^2 I)$, the PDF is given by:

$$p(\boldsymbol{y}, x) = \frac{1}{\sqrt{2\pi}^m \sqrt{|det(\sigma_n^2 I)|}} exp\left(\frac{-1}{2}(\boldsymbol{y} - \boldsymbol{a}_\theta\, x)^T \frac{1}{\sigma_n^2} I (\boldsymbol{y} - \boldsymbol{a}_\theta\, x)\right)$$

$$= \frac{1}{\sqrt{2\pi}^m\, \sigma_n} exp\left(\frac{-1}{2\,\sigma_n^2}(\boldsymbol{y} - \boldsymbol{a}_\theta\, x)^T (\boldsymbol{y} - \boldsymbol{a}_\theta\, x)\right)$$

Taking the log and then the derivative with respect to x then gives:

$$\frac{\partial}{\partial x} \ln(p(\boldsymbol{y}, x)) = \frac{-1}{2\,\sigma_n^2} \frac{\partial}{\partial x} \left((\boldsymbol{y} - \boldsymbol{a}_\theta\, x)^T (\boldsymbol{y} - \boldsymbol{a}_\theta\, x)\right)$$

$$= \frac{-1}{2\,\sigma_n^2}\, \frac{\partial}{\partial x} \left(\left(\boldsymbol{y}^T - x\ \boldsymbol{a}_\theta{}^T\right)\ (\boldsymbol{y} - \boldsymbol{a}_\theta\, x)\right)$$

$$= \frac{-1}{2\,\sigma_n^2}\, \frac{\partial}{\partial x} \left(\boldsymbol{y}^T\boldsymbol{y} - x\,\boldsymbol{y}^T\boldsymbol{a}_\theta - x\,\boldsymbol{a}_\theta{}^T\boldsymbol{y} + x^2\,\boldsymbol{a}_\theta{}^T\boldsymbol{a}_\theta\right)$$

$$= \frac{-1}{2\,\sigma_n^2} \left(-\boldsymbol{y}^T\boldsymbol{a}_\theta - \boldsymbol{a}_\theta{}^T\boldsymbol{y} + 2x\ \boldsymbol{a}_\theta{}^T\boldsymbol{a}_\theta\right)$$

Equating the derivative to zero allows to express the maximum likelihood estimator as:

$$\frac{\partial}{\partial x} \ln(p(\boldsymbol{y}, x)) = 0 \Leftrightarrow x = \frac{\boldsymbol{y}^T\boldsymbol{a}_\theta}{\boldsymbol{a}_\theta{}^T\boldsymbol{a}_\theta}$$

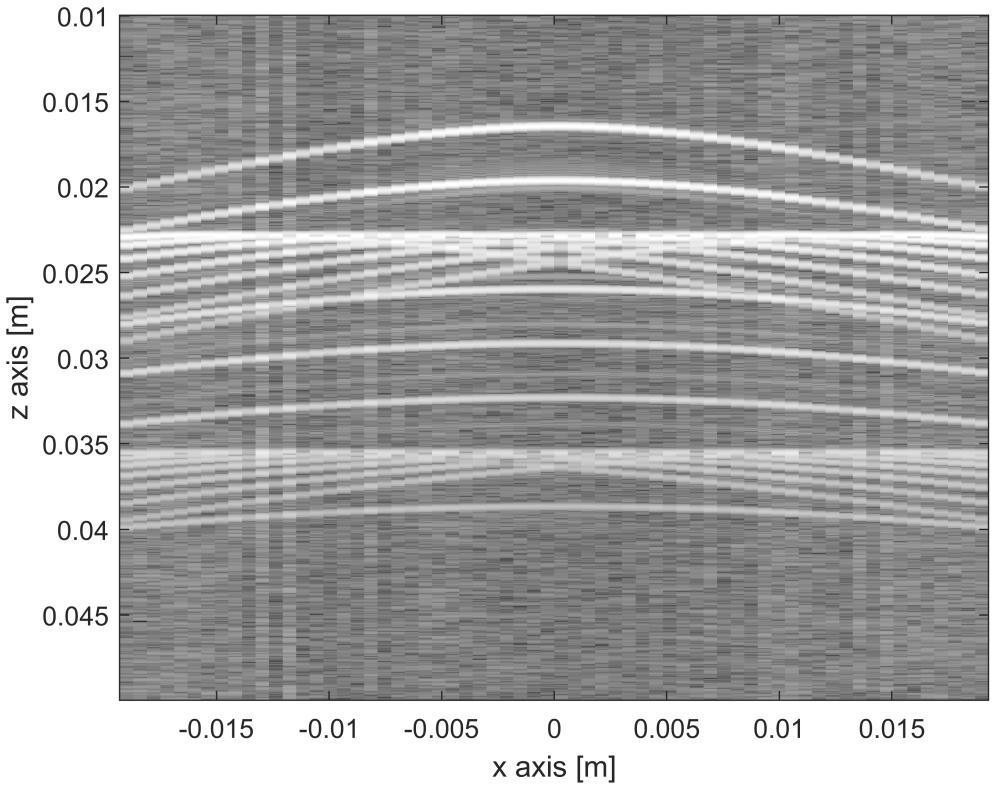
which can be simplified in this case by:

$$x = \frac{1}{m} \sum_{i=0}^{m-1} y_i$$

QUESTION 5b

Implement the ML estimator found in the previous question, and plot the reconstructed image using the envelope detection and log-compression as in question **4b** and **4c**.

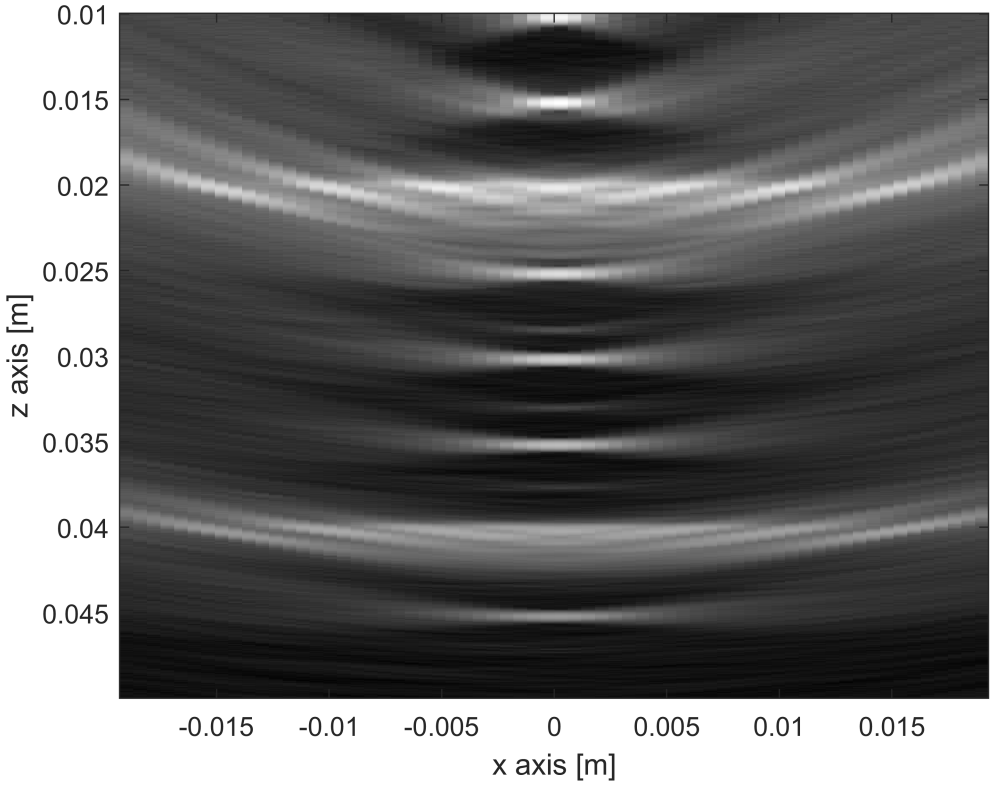
```
% Envelope detection, log compression and TOF correction
I = abs(hilbert(Y));
I = 20 * log10(I ./ max(I));
corrected_signal = tof_correction(I,x_grid,z_grid);

% ML estimator
x = mean(I, 3);

% Visualization
figure;
colormap gray; axis image;
imagesc(x_axis, z_axis, x)
xlabel('x axis [m]')
ylabel('z axis [m]')
```



QUESTION 5c

Let us now include the array response of the transducer. Assume that a_θ follows a Hanning window, and calculate the new ML estimate of x. Plot the reconstructed image, do you see any differences compared to the previous approach? Can you say something about the resolution and contrast in the image?

```
% ML estimator and visualization (~5 lines of code)
M = 64;
a = hanning(M);
x  = zeros(1301, 64);

% Envelope detection, log compression and TOF correction
I = abs(hilbert(Y));
I = 20 * log10(I ./ max(I));
corrected_signal = tof_correction(I,x_grid,z_grid);

for i = 1:1301
    for j = 1:64
        y = squeeze(corrected_signal(i,j,:));
        x(i, j) = (transpose(y) * a ) / (transpose(a) * a);
    end
end


% Visualization
figure()
colormap gray; axis image;
imagesc(x_axis, z_axis, x)
xlabel('x axis [m]')
ylabel('z axis [m]')
```



ANSWER:

The same equation as previously can be used:

$$\frac{\partial}{\partial x}\ln(p(\boldsymbol{y},x))=0 \Leftrightarrow x=\frac{\boldsymbol{y}^T\boldsymbol{a}_\theta}{\boldsymbol{a}_\theta{}^T\boldsymbol{a}_\theta},$$

which in this case can be written as:

$$x=\frac{\sum_{i=0}^{m-1}\boldsymbol{a}_{\theta_i}y_i}{\sum_{i=0}^{m-1}\boldsymbol{a}_{\theta_i}{}^2}$$

The resulting image here is less blurry, and has more clearly defined lines compared to the previous image. Additionally, the resolution seems larger (individual pixels are clearly visible in the first image), while the contrast remains similar.

Assignment 6 - Colored Noise

The white noise assumption used in the previous assignment is a very rough approximation of a realistic measurement. To improve on this, we are going to assume a colored noise profile $\boldsymbol{n} \sim N(0, C_n)$, where C_n is the noise covariance matrix.

QUESTION 6a

Derive the ML estimate for a colored noise profile.

ANSWER:

Since $\boldsymbol{y} \sim N(\boldsymbol{a}_\theta\, x, C_n)$, the PDF is given by:

$$p(\boldsymbol{y}, x) = \frac{1}{\sqrt{2\pi}^m \sqrt{|det(C_n)|}} exp\left(\frac{-1}{2}\left(\boldsymbol{y} - \boldsymbol{a}_\theta\, x\right)^T C_n^{-1}\left(\boldsymbol{y} - \boldsymbol{a}_\theta\, x\right)\right)$$

Taking the log and then the derivative with respect to x then gives:

$$\frac{\partial}{\partial x}\ln(p(\boldsymbol{y}, x)) = \frac{-1}{2}\frac{\partial}{\partial x}\left(\left(\boldsymbol{y} - \boldsymbol{a}_\theta\, x\right)^T C_n^{-1}\left(\boldsymbol{y} - \boldsymbol{a}_\theta\, x\right)\right)$$

$$= \frac{-1}{2}\frac{\partial}{\partial x}\left(\left(\boldsymbol{y}^T - x\ \boldsymbol{a}_\theta{}^T\right)\ C_n^{-1}\left(\boldsymbol{y} - \boldsymbol{a}_\theta\, x\right)\right)$$

$$= \frac{-1}{2}\frac{\partial}{\partial x}\left(\boldsymbol{y}^T C_n^{-1}\boldsymbol{y} - x\,\boldsymbol{y}^T C_n^{-1}\boldsymbol{a}_\theta - x\,\boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{y} + x^2\,\boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{a}_\theta\right)$$

$$= \frac{-1}{2}\left(-\boldsymbol{y}^T C_n^{-1}\boldsymbol{a}_\theta - \boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{y} + 2x\ \boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{a}_\theta\right)$$

Equating the derivative to zero allows to express the maximum likelihood estimator as:

$$\frac{\partial}{\partial x}\ln(p(\boldsymbol{y}, x)) = 0 \Leftrightarrow x = \frac{\boldsymbol{y}^T C_n^{-1}\boldsymbol{a}_\theta + \boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{y}}{2\,\boldsymbol{a}_\theta{}^T C_n^{-1}\boldsymbol{a}_\theta}$$

QUESTION 6b

The noise covariance matrix is generally not known, and varies across the medium and across acquisitions. A solution to this is estimating the covariance from the received data. Because the echo signals are (approximately) zero-mean we can estimate the noise covariance matrix by calculating the spatial covariance matrix $C_n \approx C_y = E[\mathbf{y}\mathbf{y}^t]$.

However, a problem that can occur with this method is that the inversion of the covariance matrix can be numerically unstable. To solve this we can apply diagonal loading, in which we add a (small) constant to the

diagonal of the covariance matrix given by $\frac{D}{M}\text{trace}(C)$, where M denotes the number of array elements, and D is a diagonal loading factor.

Estimate the noise covariance $\hat{\mathbf{C}} = E[\mathbf{y}\mathbf{y}^t]$ for each pixel (x,z) in the image, using a diagonal loading factor $D = 5$.

```
% Estimate the signal covariance per pixel
D = 5;
M = size(y);
M = M(1);
temp = size(x);
x_size = temp(1);
z_size = temp(2);
covariances = zeros(x_size, z_size, M, M);

corrected_signal = tof_correction(Y,x_grid,z_grid);

for i = 1:x_size
    for j = 1:z_size
        y_temp = reshape(corrected_signal(i, j, :), [M 1]);
        covariances(i, j, :, :) = y_temp * transpose(y_temp);
        covariances(i, j, :, :) = reshape(covariances(i, j, :, :), [M M]) + ...
            eye(M) * D * trace(reshape(covariances(i, j, :, :), [M M])) / M;
    end
end
```

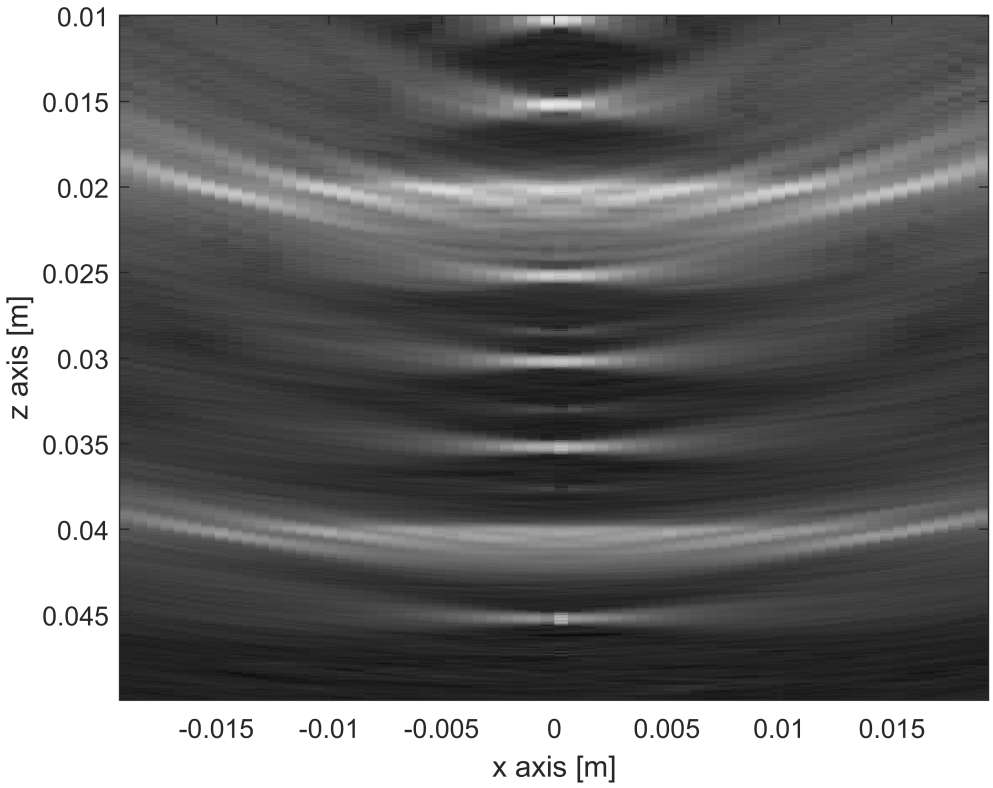
QUESTION 6c

Implement the ML estimator derived in question **6a**, and use the noise covariance derived in question **6b**.

Plot the resulting image. What differences do you observe compared to the images in question **5**.

```
% Implement the ML estimator with colored noise (~5-10 lines of code)

a = hanning(M);

I = abs(hilbert(Y));
I = 20 * log10(I ./ max(I));
corrected_signal = tof_correction(I,x_grid,z_grid);

for i = 1:x_size
    for j = 1:z_size
        y_temp = reshape(corrected_signal(i, j, :), [M 1]);
        Cn_inv = inv(reshape(covariances(i, j, :, :), [M M]));
        x(i, j) = (transpose(y_temp) * Cn_inv * a + transpose(a) * Cn_inv * y_temp) / ...
            (2 * transpose(a) * Cn_inv * a);
    end
end
```

```
% Visualization
figure()
colormap gray; axis image;
imagesc(x_axis, z_axis, x)
xlabel('x axis [m]')
ylabel('z axis [m]')
```



QUESTION 6d

We have now used a relatively small diagonal loading factor. What would happen if we select a value that is (too) high? Can you comment on the assumed noise model?

ANSWER:

If diagonal loading factor is too high the covariance matrix looks too similar to a diagonal matrix. That means that the information of cross-covariance between each pixels is lost because is much smaller than the diagonal elements of the covariance matrix.

The colored noise model gives a better space resolution to the image providing clear edges. This model results more computationally expensive due to the covariance matrix calculations and its inverse.