KTH Matematik
Mathematical Statistics

SF1900/SF1912 Probability and Statistics, Autumn 2025
Computer exercise 1 for TCOMK3/CINTE3

1 Introduction

This computer exercise is a home assignment and can not render you bonus points on the final written exam and re-exam. It is intended as a preparation for the second computer exercise, which can render you bonus points on the written exam. Its purpose is to help you get acquainted with the statistical functions in MATLAB, and to give you a deeper understanding of some of the basic concepts of the course, by illustrating them using MATLAB. You can do the exercise whenever you like, but we recommend to work on it before the short written exan on week 38. The results of Lab 1 do not have to be submitted on canvas. Try to focus on learning how to use MATLAB and make sure that you understand the commands you use.

2 MATLAB

In both of the computer exercises in this course we will use MATLAB, which is an interactive program for numerical computations. It is also a programming language. You will find MATLAB installed on most of the computers at KTH, and one of the advantages with MATLAB is that it will look basically the same regardless of the type of computer, on which you run it. If you would like to read more about MATLAB there are several guides to download or buy.

Start by logging on to your usual account. Then start MATLAB by clicking on the icon. The program is terminated by typing the command `exit`. To begin with you can think of MATLAB as an advanced calculator which evaluates expressions. You type in what you want to be done and MATLAB answers.

```
>> 3*11.5 + 2.3^2/4
ans =
      35.8225
```

Variables are assigned values using the sign `=`, and these values will be stored in the memory.

Try to assign values to some variables.

```
>> a = 1;
>> b = sqrt(36);
>> width = 3.89;
>> who

Your variables are:

a          b          width
```

As you can see the command `who` shows a list of variables that have been stored in the memory. A variable, for instance `b`, can be erased from memory using `clear b`. If you put a semi-colon , `;`, at the end of a row, the result of the commands on that row will not be written on the screen. MATLAB can also handle vectors and matrices, just as easily as scalars.

```
>> x = [1 3 7]

x =

     1 3 7

>> y = [2 1 8]'

y =

      2
      1
      8

>> z = [1 2 ; 3 4]'

z =

      1   3
      2   4

>> w = rand(1, 4)

w =

     [0.2190    0.0470    0.678    0.6793]
```

The sign `'` translates to transpose, as you can see from the above, and the semi-colon is used to separate rows in matrices. The function `rand(m, n)` will generate an $m \times n$-matrix of random numbers uniformly distributed between 0 and 1. The notation in algebraic expressions is the usual one, but you should keep in mind that the sign for multiplication `*` is interpreted as matrix multiplication. Element-wise multiplication of two matrices `A` and `B` of the same dimensions is written `A.*B`. MATLAB has most common functions built such as

```
exp log sin asin cos acos tan atan
```

Please observe that `log` denotes the natural logarithm. Try to plot a function, for instance by executing the following commands.

```
>> x = 0.5:0.1:2
>> help log
>> y = log(x)
>> plot(x,y)
```

In the first row `x` is assigned a vector running from 0.5 to 2 in steps of 0.1. The help function `help` will give you information about a function if you type `help` and the name of the function. If you only type `help` a list of all available functions will be shown sorted by *toolbox*).

MATLAB contains a great number of functions related to probability theory and statistics. See `help stats`. Look in particular at what you find under the heading *Random Number Generators*, which you will find at the beginning of the long list of functions.

For this computer exercise you will need, in addition to the functions already in MATLAB, the two functions listed below. They can be download from the course page in Canvas.

- `plot_mvnpdf.m`
- `hist_density.m`

Make sure the files are downloaded to the directory you will be working in. To make sure the files are in the right directory type `ls` to list the files in the current working directory.
You can write your commands directly at the prompt in MATLAB, but usually it is easier to work in the editor. If the editor is not open you can open it and create a new file by typing `edit lab1.m`. The code included below is written in sections. A new section is begun by typing two percent signs. The code `Ctrl+Enter` executes the commands in a section.

3 Simulation

The theme for this computer exercise is simulation. In the part of the course which treats probability theory you will learn how to compute different quantities, such as probabilities, expected values etc., given a certain probability distribution. For more complicated random models it can be very time-consuming, or even impossible to do exact computations.

In such circumstances simulation can be an alternative. You then use a computer to simulate outcomes from a certain distribution, and then you can use for instance the mean of the outcomes to estimate the expected value, or some empirical quantile to estimate a probability. In this computer exercise we will do this for some rather simple problems (where explicit computations are possible), but the same basic principles can be applied to more difficult problems where explicit computations are not possible.

Let us say that we want to estimate the expected value of a certain distribution. Suppose that the distribution function is F, and let X be a random variable with this distribution function. Also let us denote the expected value we want to estimate by μ. If we generate observations $x_1, x_2, \ldots, x_n$ from F, we can estimate μ using

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k. \tag{1}$$

This is a reasonable estimate of μ due to the law of large numbers. If we instead want to estimate $F(a) = \mathbb{P}(X \leq a)$ for some number a, then this can be done by calculating the proportion of the simulated observations that satisfies $\leq a$. We can write this as

$$\hat{F}(a) = \frac{\text{number of } x_k \leq a}{n} = \frac{1}{n} \sum \mathbb{I}(x_k \leq a). \tag{2}$$

Here $\mathbb{I}$ denotes a so-called *indicator function*, which takes the value 1 if the condition within parentheses is fulfilled, and 0 otherwise. This means that $\mathbb{I}(x_k \leq a)$ is equal to 1 exactly for those k such that $x_k \leq a$a, which in turn means that the sum above counts the number of indices k which fulfill the condition.

4 Computer exercises

Problem 1 - Simulating exponentially distributed random numbers

The function in MATLAB used to simulate exponentially distributed random variables is called `exprnd`. Please use the help command `help` to find out exactly how the function `exprnd` works, i.e. type `help exprnd`. Note that the MATLAB function `exprnd` uses the expectation μ as the parameter in contrast to [2] which uses $1/\mu$ as the parameter. The function can also be used to simulate vectors (or matrices) of independent exponentially distributed random variables.

The following code generates N random numbers from the distribution Exp(1/10), draws a histogram of the simulated data, and finally plots the true density function of Exp(1/10) in the same figure as the histogram for comparison.

```
%% Problem 1: Simulating exponentially distributed ...
    random numbers
mu = 10;
N = 1e4;
y = exprnd(mu, N, 1); % Generates N Exp(mu) random numbers
hist_density(y);      % Creates a normalized histogram
t = linspace(0, 100, N/10); % Vector of N/10 points
hold on
plot(t, exppdf(t, mu), 'r') % 'r' means red line
hold off
```

Redo the simulation and study how the histogram changes. What is the relation between the red line and the histogram, and how would you explain the fluctuations around the red line?

Problem 2 - Law of large numbers

If $X_1, X_2, \ldots, X_n$ are identically distributed with expectation μ then according to the law of large numbers in [2] it holds that

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| < \varepsilon\right) \to 1 \tag{3}$$

for every $\varepsilon > 0$ when $n \to \infty$. This means that the probability that the difference between the true value of the expectation and the mean value is less than ε tends to one as the number of observations tends to infinity. We will now look at this convergence by simulating random variables X_i (which will be exponentially distributed here) and then studying the behaviour of the mean $S_n = (X_1 + \cdots + X_n)/n$.

```
%% Problem 2: The law of large numbers
mu = 0.5;
M = 500;
X = exprnd(mu, M, 1);
plot(ones(M, 1)*mu, 'r-.')
hold on
for k = 1:M
    plot(k, mean(X(1:k)), '.')
    if k == 1
        legend('True \mu', 'Estimated \mu')
    end
    xlabel(num2str(k)), pause(0.001)
end
hold off
```

If you get tired of waiting you can comment out the row with `pause` by typing a percent sign at the beginning of the row. The point corresponding to the value Does the plot look the way you expected it to?

Problem 3 - Using Monte Carlo simulation to compute an expectation

Now, suppose that we did not know that the exponential distribution that we simulated random numbers from in Problem 1 had an expectation of 10, and that we would like to estimate the expectation using only the simulated data. We can do this by computing the average of the simulated data.

```
%% Problem 3: Expectation of exp distributed rv
    N = 1e4;
    y = exprnd(mu, N, 1);
    mean(y);
```

How good is your estimate? Try redoing the simulation and averaging a few times. Also try different values of `N`.

To use the mean of simulated random numbers to (approximately) compute an expectation is known as the Monte carlo method. The idea behind the Monte Carlo method is old and can be found in mathematics from the 18th century, but it was not until the second half of the 20th century, when computers made it possible to do massive amounts of computations, that the modern versions of the method were developed. In the late 1940s Stanislaw Ulam and John von Neumann developed methods to make the "throw of a dice" using a computer, see [1]. The method has been named after the Monte Carlo Casino in Monaco, and was used in the simulations for the Manhattan Project, a research and development project that produced the first nuclear weapons.

Problem 4 - Monte Carlo simulation of the number π

An advantage of using the Monte Carlo method to compute an expectation is that it can be used to compute expectations which are difficult to compute analytically. We will now use Monte Carlo methods to determine an approximate value of the number π. Let U and V be two independent random variables which are uniformly distributed on the interval $[-1, 1]$. The pair (U, V) assumes values in $[-1, 1] \times [-1, 1]$ and can be seen as a point in this square in the plane. The probability that the point (U, V) ends up in the unit circle is

$$P(\sqrt{U^2 + V^2} \leq 1) = \frac{\text{area of unit circle}}{\text{area of square } [-1, 1] \times [-1, 1]} = \frac{\pi}{4}.$$

We can estimate π in the following way. First we simulate a large number of points $(U_1, V_1), (U_2, V_2), ..., (U_N, V_N)$. For each point (U_i, V_i) we check if $\sqrt{U_i^2 + V_i^2} \leq 1$ and compute the proportion of points within the unit circle, to the total number of points. Since

$$\frac{\text{The number of points within the unit circle}}{N} \to P(\sqrt{U^2 + V^2} \leq 1) = \frac{\pi}{4},$$

as $N \to \infty$, we have that for large values of N

$$\pi \approx \frac{4 \cdot \text{The number of points within the unit circle}}{N}.$$

The following code generates N points (U_i, V_i), plots them in the plane, and computes an estimate of π. Run the code several times and use different values of `N`.

```
1  %% Problem 4: Monte Carlo estimate of the number pi
2      N = 1e2;
3      U = 2*rand(1,N)-1;  % Generates random numbers from U(-1,1)
4      V = 2*rand(1,N)-1;
5      plot(U,V,'o'), hold on  % Plots the simulated points
6      X = -1:0.01:1;
7      plot(X,sqrt(1-X.^2),'r')    % Plots unit circle
8      plot(X,-sqrt(1-X.^2),'r')
9      Z = (sqrt(U.^2+V.^2)<=1);    % Computes approximate ...
           value of pi
10     pi = 4*mean(Z);
```

Note that the command `(x>5)` in MATLAB will return a vector of the same size as `x`, but with elements that are 1 or 0 depending on whether the corresponding element in `x` satisfies the condition `> 5` or not. This is used in row 9 in the code above.

Problem 5 - Computation of probabilities

MATLAB has commands for the most common probability distributions. Read the help files for the functions `binocdf`, `binopdf`, `normcdf`, `normpdf`, `expcdf`, and `exppdf`. Note that, just as `exprnd`, `expcdf` and `exppdf` use the expectation μ as a parameter. Let $X_1 \in \text{Bin}(10, 0.3)$, $X_2 \in N(5, 3)$, $X_3 \in \text{Exp}(7)$ and determine (using the MATLAB functions above) for each $k = 1, 2, 3$,

1. $P(X_k \leq 3)$
2. $P(X_k > 7)$
3. $P(3 < X_k \leq 4)$

Problem 6 - Visualizing probability distributions

There are commands in MATLAB to help visualize the most common probability distributions. The value of the probability density function of the normal distribution $N(\mu, \sigma)$ in the point x is can be computed using the command `normpdf(x,mu,sigma)`. The following code generates the graph of the probability density function of the standard normal distribution $N(0, 1)$.

```
%% Problem 6: Pdf of standard normal distribution
    dx = 0.01;
    x = -10:dx:10;      % Creates a vector with increments ...
        of length dx
    y = normpdf(x,0,1);
    plot(x,y)
```

Try plotting the probability density function of some other normal distributions $N(\mu, \sigma)$, for example $\mu = -1$, $\sigma = 0.1$ and $\mu = 2$, $\sigma = 2$, respectively.

The gamma distribution with parameters a and b has a probability density function given by

$$f_X(x) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-x/b},$$

(please note that in [2] another parameterization of the gamma distribution is used). The following code can be used to plot the density function of the gamma distribution.

```
%% Problem 6: Pdf of gamma distribution
    dx = 0.01;
    x = 0:dx:10;        % Creates a vector with increments ...
        of length dx
    y = gampdf(x,1,2);
    plot(x,y), hold on
    z = gampdf(x,5,1);
    plot(x,z,'r')
```

There are commands also for the (cumulative) distribution functions of the most common probability distributions. The following code can be used to plot the distribution function of a gamma distribution.

```
%% Problem 6: Cdf of gamma distribution
    dx = 0.01;
    x = 0:dx:10;        % Creates a vector with increments ...
        of length dx
    y = gamcdf(x,1,2);
    plot(x,y), hold on
    z = gamcdf(x,5,1);
    plot(x,z,'r')
```

Problem 7 - Multivariate normal distribution

The probability density function of the multivariate normal distribution can be visualized using `plot_mvnpdf`. Investigate what the function does and try some different values of the parameters. The parameters `mux` and `muy` can take on any real value, whereas the parameters `sigmax` and `sigmay` can only take on positive values, and the parameter `rho` can take values in the interval $[-1, 1]$. Please observe that the axes of the plot are fixed in the function `plot_mvnpdf` so for parameter values that are greater than, or less than ten most of the density function will be outside the range of the plot.

```
%% Problem 7: Multivariate normal distribution
    mux = 0; muy = 100; sigmax = 1; sigmay = 4; rho = 0.7;
    plot_mvnpdf(mux, muy, sigmax, sigmay, rho)
```

How does changing the parameter values affect the plot? What is the interpretation of the different parameters?

Referenser

[1] Eckhardt, Roger (1987) Stan Ulam, John Von, Neumann, and the Monte Carlo, Method *Los Alamos Sci.*, Vol **15**, p. 131-43.

[2] Blom, G (1989). Probability and Statistics. Theory and Applications.