

Computer exercise 1

Introduction to Matlab.

Understanding Distributions

Please write your names and "personal identification numbers" here. During the exercise fill in the blanks marked by black bullets and answer the posed questions. To pass the exercise, all questions should be answered and handed in to the computer exercise supervisor.

•

All necessary files are downloadable from the course home page

<http://www.math.chalmers.se/Stat/Grundutb/CTH/mve300/0910/files/data.zip>.

Please download the data.zip file and uncompress it at the directory you plan to use for the computer exercises.

In this exercise, you will first be given an introduction to Matlab, which is an integrated technical computing environment. Matlab will be used in the computer exercises throughout this course. We proceed by exploring the concepts of probability and distributions intuitively by means of numerical examples in Matlab.

1 Preparatory exercises

1. Read the instructions for the computer exercise and chapter 3.1-3.3 and 4-4.1 in the book .
2. Make sure you understand what probability and density functions are and how they are related to the distribution function.
3. Given a sample $\{x_1, \dots, x_n\}$ from a random variable (r.v.) X , how do you construct the empirical distribution function? What is the empirical distribution function?

•

4. Explain what is meant by the α -quantile of a distribution.

•

2 Matlab – the first steps

Matlab¹ allows the user to combine numeric computation with advanced graphics and visualisation. Short commands can be executed interactively, but for more complicated problems, it is also possible to perform programming, defining own functions &c. In addition to Matlab, several so-called toolboxes exist for specific applications, like signal processing, control theory, finite-element methods. In the computer exercises, we will make use of, among others, the commercial Statistics Toolbox and the freely redistributable WAFO toolbox².

Use of matrices and vectors in Matlab

Matlab can be used as an advanced calculator; the most common functions are predefined. At the Matlab prompt (`>>`), you can for example calculate $\sqrt{1,19^2 - 1} + \sin(\pi/2) + e^2$ by typing

```
>> sqrt(1.19^2-1)+sin(pi/2)+exp(2)
```

and the result appears on the screen. When you want to find out more about predefined functions in Matlab, the help-command `help` is useful. It is a good rule to make use of it during the exercise, even if not explicitly stated in the text! First, write `help help`. As an example, write `help log` to find out which base Matlab is using as default in the logarithm function.

Matlab is shorthand for Matrix laboratory, and use of vectors and matrices is characteristic for Matlab. All data are stored in vectors or matrices. (With a vector, we mean a row or column matrix.) The matrix

$$A = \begin{pmatrix} 2 & 0 \\ 3 & 1 \end{pmatrix}$$

is entered in the following way:

```
>> A=[2 0; 3 1]
```

An example of a vector is given by

```
>> v=[0 0.1 0.2 0.3];
```

(A semicolon after a written statement prevents the echo on the screen, and may be useful if long vectors are entered.) Now, there is a trick to build up vectors in an easy way: the vector `v` can also be defined by typing

```
>> v=0:0.1:0.3
```

The command `length` (or `size`) determines the size of the vector (or matrix):

```
>> vLength=length(v)
>> ASize=size(A)
```

It goes without saying that accessing elements in a vector is a very important act. Suppose you want the value of the fourth element in the vector, as well as the values of the first three elements. The solution is given by

```
>> v(4), v(1:3)
```

¹More information about Matlab is found at <http://www.mathworks.com/>

²WAFO = Wave Analysis for Fatigue and Oceanography. The WAFO toolbox is put together by the WAFO group based at Lund University. See <http://www.maths.lth.se/matstat/wafo/>

or, together,

```
>> v([4 1:3])
```

(Note that statements can follow consecutively, separated by commas or semicolons.) Elements in a vector can be sorted in ascending order:

```
>> u=[8 -3 2.7]; uSorted=sort(u)
```

Handling variables and data

We have defined a number of variables, and a list of current variables is given by typing `who`. The command `whos` is similar, but does also return the size of the variables. Try these commands; do you recognise the names of the variables? We close this section by removing the objects. All variables are removed by just typing the `clear`. Try `help clear` to find out how to just remove specific variables.

If a worksheet in Excel is saved on disk as a tab separated file, one can import it to the Matlab workspace: For example, a data sheet, stored as `Box1.txt` can be read into Matlab by entering `load Box1.txt -ascii` at the Matlab prompt. The data in `Box1.txt` will then be loaded, and afterwards, in Matlab's workspace, the data is referred to as `Box1`, i.e. the file name without its extension. This works only on condition that, firstly, the file `Box1.txt` contains only numerical values, and, secondly, all rows have the same number of elements, and, thirdly, the decimal sign is a full stop (not a comma). Much more general and convenient is command `xlsread`; type `help fileformats` for more information. If you want, you can open Excel, enter up some data, save it (see that you get appropriate file format), and try to load it into the workspace of Matlab.

Graphics and visualisation

In this section, we will see how to make simple plots in Matlab. After you have studied it, you will know how to make a plot of a function $x \mapsto f(x)$. As an example, let us choose $f(x) = \sin x$, $0 < x < 4\pi$.

If vectors are defined, they may be visualised by the command `plot`. We first define vectors `x` and `y` as

```
>> x=0:0.05:4*pi; y=sin(x);
```

Use the command `length` as you learnt previously, to find out the length of `x` and `y`, respectively. Two vectors of the same length can be plotted against each other. If you type

```
>> plot(x,y)
```

a graphical window will appear, and a plot will be drawn. The figure is referred to as `figure(1)`. It is possible to have a number of graphical windows accessible.

Several options can be given to the `plot` command. One example is colour:

```
>> plot(x,y,'r')
```

Another characteristic is the plot symbol. If we plot the values marked as stars, we are reminded about the definition of the vectors in the computer; they are composed of a number of discrete points.

```
>> plot(x,y,'*')
```

You can combine plot options. Read `help plot` and find out what the following command will perform, then check it on the screen:

```
>> plot(x,y,'md-')
```

The `axis` command may sometimes be useful to display interesting regions in a figure; try

```
>> axis([0 10 -1.5 1.5])
```

A plot is most often easier to study if a grid is inserted: try to find out how to use the command `grid` and then apply a grid to the current plot.

The current figure is deleted by means of the command `clf`. An empty window will remain on the screen after this operation. If you want also the window to disappear, then use the command `close` instead.

3 Relative frequencies and distributions

In this section, we will use numerical examples in Matlab to approach the concepts “probability” and “distribution”. The aim is that you should obtain an intuitive feeling for probabilistic reasoning, rather than to be immediately confronted with theory.

Exploring data

For illustrational purposes, we will use artificial data, which are simulated from a statistical distribution. This is opposite to real-world data, where no labels, explaining the statistical distributions, are found. However, although we, statistically speaking, know the origin of the data, this approach is useful from a pedagogical point of view. Even in research on statistical-computation algorithms, simulated data are often used for analysis and testing.

To obtain a random data-set of 50 values, type

```
>> data=randn(1,50);
```

What is the distribution of your random sample (use `help randn`)? Write down the density function.

•

A good rule, whenever a new set of data is encountered: try to plot it in some kind of diagram! Use the plot command: `plot(data, '.-')`. Another way of presenting the data is to plot the sorted data: `plot(sort(data), '.-')`. From the data set got above, choose a relatively high number, say, $x = 1.1$. It may be interesting to calculate the percentage of data which have values less than or equal to this number. When the number of observations in the sample increases, we may interpret the ratio as the probability to obtain a value less than x . The ratio is calculated as follows:

```
>> x=1.1; ratio=sum(data<=x)/length(data)
```

(See that you understand the commands!) Try some other values of x . How do you expect the percentage to change? Compare with the plot with sorted data.

•

The opposite procedure, that is, find the value x corresponding to a given probability, is often more important. This is referred to as finding the quantiles. We will return to this later.

We can of course let the computer choose a large number of values x to examine, and then try get an overview. This is implemented in a home-written function `empcdf`. The function delivers two vectors: `x` contains the values chosen, while the ratios are collected in `ratio`. (If you want to see the code, try `type empcdf`.) The result is visualised in a new figure:

```
>> [x,ratio]=empcdf(data);
>> figure(2);
>> plot(x,ratio,'.')
>> grid on
```

The figure should look similar to Figure 1 in this paper. It shows in some sense how the values in data are distributed, and the resulting function is called the empirical distribution function³. For a value on the abscissa, say, 1.1, we find the percentage of values in the sample with values less than this number.

Another way to plot the empirical distribution function is to make use of the command `stairs`:

```
>> n=length(data);
>> figure(3);
>> stairs(sort(data),(1:n)/n)
>> grid on
```

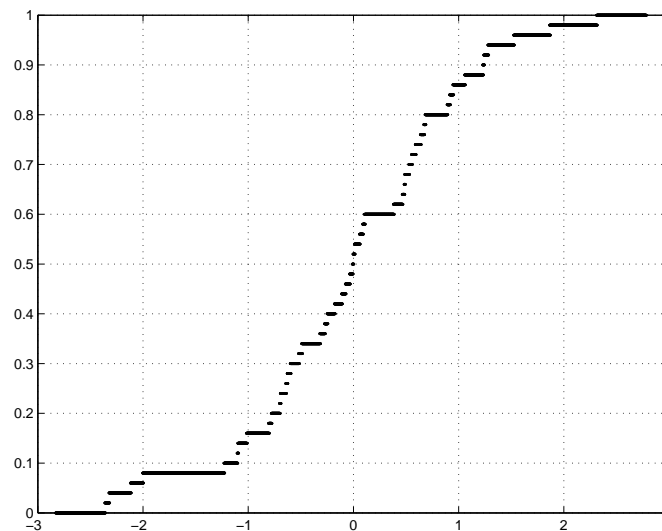


Figure 1: Empirical distribution function, an example.

Larger samples. The distribution function of a random variable

Let us now study another sample of 50 observations, obeying the same random law as the previous data. We simulate data, and plot them in the same figure as before:

```
>> data=randn(1,50);
>> [x,ratio]=empcdf(data);
>> figure(2);
>> plot(x,ratio,'.')
>> grid on
```

³Distribution functions are often called cumulative distribution functions; that is why our home-written routine is called `empcdf`, empirical cumulative distribution function.

Continue with a larger set of data, say, 2000 observations. Analyse them as before:

```
>> data=randn(1,2000);
>> [x,ratio]=empcdf(data);
>> figure(2);
>> plot(x,ratio,'.')
>> grid on
```

With a large number of observations, the result approaches the distribution function, that is, for a random variable X , the function

$$F_X(x) = P(X \leq x). \quad (1)$$

In our case, X was chosen from a Gaussian distribution (normal distribution); we had that $X \in N(0;1)$. It is instructive to plot the theoretical distribution function, implemented in `normcdf`, in the same figure as before:

```
>> figure(2)
>> hold on
>> plot(x,normcdf(x),'r')
>> hold off
```

For every distribution function F_X , we have that $F_X(x) \rightarrow 1$ when $x \rightarrow \infty$ and that $F_X(x) \rightarrow 0$ when $x \rightarrow -\infty$. Interpret the figure! What are on the x and y axis? Estimate the median. Is it possible to estimate the mean of the distribution from the plot?

•

Quantiles

The concept of quantile (or fractile), mentioned before, is important. The quantile can be defined in different ways – when studying tables &c., you should always make sure of which definition is used. We here define the quantile as a number x_α which satisfies

$$P(X \leq x_\alpha) = 1 - \alpha \quad (2)$$

where α is some small number (common choices: 0.05, 0.01, 0.001). The quantile is not always unique; for some values of α there might be infinitely many x_α satisfying (2); for other values of α there might be no quantile x_α at all. From your Matlab-plot (`figure(2)`), using (1) and the definition of quantile, can you estimate the quantile $x_{0.05}$ when $\alpha = 0.05$?

• $x_{0.05} =$

Compare with the exact value, given by `norminv(1-0.05)`.

Other distributions

Some common choices of distribution functions have their own names. They are not only just functions in a mathematical sense, but have also been found suitable when modelling random phenomena in science and technology. The distributions are listed in almost any basic text-book in mathematical statistics.

Some of the distributions are implemented in the Statistics Toolbox. You have already encountered the distribution function when $X \in N(0; 1)$; it is often denoted by Φ :

$$F_X(x) = \Phi(x) = \frac{1}{(2\pi)^{1/2}} \int_{-\infty}^x e^{-t^2/2} dt$$

An easy way to obtain new distributions is scaling random variables or adding constants to them: Suppose that the distribution function $F_X(x)$ of some stochastic variable X is known. If a new stochastic variable Y is defined as $Y = aX + b$, where $a > 0$ and b are constants, we can perform the following calculation

$$F_Y(y) = P(Y \leq y) = P(aX + b \leq y) = P(X \leq (y - b)/a) = F_X((y - b)/a)$$

to obtain the distribution function for Y . What happens if $a < 0$ or if $a = 0$?

- $F_Y(y) =$

This transformation is governed by two parameters a and b ; other distributions or transformations of distributions might be governed by other sets of parameters. When analysing real-world data, one often knows from experience which type of distribution is suitable to describe the data. What remains is then to try to estimate the parameters out of data.

The normal distribution, for instance, is characterised by two parameters, m and σ^2 (or σ): if X is standard-normal, i.e. $X \in N(0; 1)$, then for $Y = \sigma X + m$ we have that $Y \in N(m; \sigma^2)$, where m and σ^2 (or σ) are the parameters.

The Gumbel distribution and the Weibull distribution

Two important distributions, with which we will meet up again in the course, are the Gumbel distribution (also called type I extreme value distribution, or double exponential distribution) and the Weibull distribution. A Gumbel distributed random variable X has the distribution function

$$F_X(x) = \exp(-e^{-(x-b)/a}), \quad -\infty < x < \infty.$$

Here, b is a location parameter and $a > 0$ is a scaling parameter. If X belongs to a Weibull distribution, we have

$$F_X(x) = 1 - \exp(-((x - b)/a)^k), \quad x \geq b \tag{3}$$

where k is a shape parameter, b is a location parameter and $a > 0$ is a scaling parameter. How does $F_X(x)$ behave when $x < b$?

-

Let us make some plots of these distribution functions, and let us use routines from the WAFO toolbox. Try the following two cases of a Gumbel distribution:

```
>> help wgumbcdf
>> figure
>> x=-4:0.05:6;
>> a=2; b=0; F1=wgumbcdf(x,a,b);
>> a=1; b=1; F2=wgumbcdf(x,a,b);
>> plot(x,F1,'b',x,F2,'r')
```

Two examples of a Weibull distribution are drawn by typing

```
>> help wweibcdf
>> figure
>> x=linspace(0,6,200);
>> a=1; k=1; F1=wweibcdf(x,a,k);
>> a=2.3; k=1.8; F2=wweibcdf(x,a,k);
>> plot(x,F1,'b',x,F2,'r')
```

Note that the WAFO routine `wweibcdf` models only the case when $b = 0$, cf (3).