

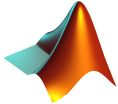
MATLAB Introductory Course

Computer Exercise Session 1

This version: January 11, 2010

1 Welcome to MATLAB

This session introduces some important parts of MATLAB— but you will not get far without searching more information on your own. Therefore, use the MATLAB help system or the book [1] to find out more about MATLAB.

To log on to the computers at ISY you need a student account and a password. Having logged in, you can start MATLAB¹ through the icon  . A

MATLAB window will open. It contains three parts:

- **Workspace:** Here the variables that you have entered will be listed (none so far).
- **Command History:** Recently run commands.
- **Command Window:** The most important window. This is where you enter the commands to run MATLAB.

2 Preparatory exercises

Before the first lesson you should do the following:

¹Depending on operating system, there may be different ways of starting MATLAB. Ask your teaching assistant if you run into problems.

1. Read the introduction chapter: “What is MATLAB”, which you can find (as a link) if you open helpdesk (write helpdesk at the command line — see Section 3).
2. Start working with Sections 3–4. Complete as much as you can of the exercises. Make sure you understand what you are doing and what the commands are doing. Collect your questions and bring them to the lesson, to ask the assistant.

3 The MATLAB help system

The MATLAB help system is reached by entering

```
>> helpdesk
```

in the command window.

A window containing the MATLAB help will open. The left part of the window contains a navigator to help you browse through the documentation. There are different ways to get help. A thorough introduction to MATLAB is found in “Getting started with MATLAB”. It contains the topics covered in this course and some additional material.

Apart from “Getting started ...”, the help system contains an index of the MATLAB commands, a search function, and demos in the help system.

The command

```
>> help command
```

prints a short help text on the command `command` (`command` should be replaced by the command you would like to know more about). It is a good habit to use `help` on all new commands you run. For instance, try it on itself!

Another very useful command is `doc`. Find out what it does.

You can also find more help in the book [1].

The best documentation is often the one you have written yourself. Therefore, make a list of handy commands as you encounter them during the exercise sessions.

4 Matlab as a calculator

By entering numerical expressions, MATLAB can be used as a calculator. Try

```
>> 5+3*2-4/7
```

What (and why?) is the result of

```
>> round(10*ans)/10
```

.....

Other round off commands include `floor` and `ceil`. Find out what they do.

.....

.....

.....

What is `ans`?

The up and down arrows of the keyboard can be used to browse the commands you have entered previously. Try this to compute

```
>> 5+3*2-4/7
>> round(10*ans)/11
```

You may also enter the first letter(s) of a command that you have already entered. The arrow keys will then browse through all entered commands that start with these letters.

Elementary functions

Compute

$\cos(\pi/3) = \dots\dots\dots \sin(\pi/4) \dots\dots\dots$

In some cases, one has to take into account that MATLAB has a limited precision. What should be the result of

```
>> sin(pi)
```

Other related commands:

Exponential functions can be entered by using `^`. To use the base e there is a special command, `exp`.

The imaginary unit is denoted `i` or `j`. Some commands that are relevant to complex numbers are `imag`, `real`, `abs`, and `angle`.

Display format

Normally, MATLAB displays floating point numbers with 4 decimal digits. If you would like to see more decimals, you can use

```
>> format long
```

To switch back, use `format short`.

If you don't want to see the result of a command, add a semicolon at the end of the expression.

Variables

For more complex computations, it is often convenient to store intermediate results in variables:

```
>> a=3
>> b=6;
>> c=b*a
```

The defined variables are visible in the “Workspace” window. You could also use the command `whos`.

If you use the same variable name again, it gets a new value. For instance, you can write

```
>> a = a+1
```

To delete variables, there is a command named `clear`. The risk of making mistakes often decreases if you clear variables when they are no longer needed. For large computations, it is sometimes also necessary, to avoid filling the entire memory.

5 Scripts and documentation

Often, you would like to run a sequence of commands several times, perhaps with different values of the variables. This could be done through MATLAB-scripts, or m-files. An m-file is simply a text file with a sequence of MATLAB commands, with the file extension `.m` (the same extension is also used for functions, which we will study later on). The file name will become a command name that can be used in MATLAB like any other command.

To remember what the script does, and thus be able to use it another day, you should include comments in your code. Comments start with the sign `%`.

Start MATLAB's editor with the command `edit`. (Other text editors can also be used — pick your favourite.) Write some code in your m-file:

```
disp('The script is running!');
stud1 = 3; % Grades for student 1 och 2
stud2 = 5;
mean = (stud1 + stud2)/2; % The average grade is computed
disp('Result:');
display(mean);
```

Save the script by choosing `Save` or `Save As` in the `File` menu. Name the script `myscript.m`. The script is saved in the file `myscript.m`. If the script is saved in a certain directory (folder), it might be necessary to change to this directory in MATLAB to be able to run it. The command `dir` prints the contents of the current directory (where you are), the command `cd` can be used to change directory, and `pwd` shows in which directory you are.

Now, you can use it just like other MATLAB commands:

```
>> myscript
```

Suppose that student 1 complained about the grading and got the grade 4 instead. Make the appropriate changes to the script and run it again. Remember to save it before running.

6 Matrices

Matrices is the basic data type in MATLAB, and there are very powerful built-in functions for matrix manipulations and computations.

Construct some matrices:

```
>> A = [1 2 5;3 8 10]
>> b = [7; 4; 5]
>> c = [3 2 1]
```

What is the function of semicolons here?

What do the following commands do?

```
>> b(3) ? .....
>> A(2,3) ? (What is the meaning of 2 and 3?) .....
>> A(:,2) ? (Here : means "all rows".) .....
>> A(1,:) ? .....
```

To retrieve elements from a matrix in this way is called *indexing*.

We can perform matrix calculations:

```
>> A*b .....
```

Why wouldn't A*c work?

Use .' to get the matrix transpose. (Using ' gives the conjugate matrix.) Try it out on some matrix!

There are other ways to create certain special matrices. What do the following commands do:

```
>> H = ones(3,2) ? .....
>> B = eye(3) ? .....
>> y = 3:9 ? .....
>> x = 1:4:21 ? (What is the purpose of the 4 here?) .....
>> z = 10:-0.5:7 ? .....
```

A few more examples of indexing — what happens here?

```
>> ind = [1 4 5]
>> y(ind)
>> A(2,2:3)
```

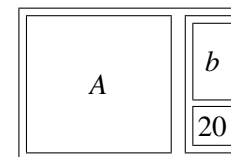
Indexing submatrices also works for assignments:

```
>> A = ones(4,4)
>> A(1,1) = 9
>> A(2,3) = 4
>> A(3,2:4) = [7 3 9]
```

Construct a larger matrix from other matrices and vectors:

```
>> A = ones(4,4)
>> D = [A [b;20]]
```

The brackets are used here to group a number of submatrices into a new matrix. As before, the semicolon indicates a line break (cf. the figure below).



Why doesn't $D = [A \ b; 20]$ work?

How would you add a row of ones below D , i.e., create the matrix

$$E = \begin{bmatrix} D \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

.....

.....
.....
.....
.....
.....

Matrix functions

Most elementary functions work with matrices, too, where they operate componentwise. Operations that have a special meaning in connection with matrices (such as $*$, $/$, and \wedge) can be forced to operate componentwise by adding a point before the operator. Try for instance

```
>> x = 0:0.1:1
>> cos(x).^2./x
```

What is the difference between

```
>> A*B
```

and

```
>> A.*B
```

(try it on some suitable matrices)?

.....

Write a script that verifies the identity $\sin^2 x + \cos^2 x = 1$ by componentwise operations for $x = 0.5, 1, 1.5, 2, 2.5, 3, 3.5$ and 4. What will your code look like?

A common application for matrices is to represent systems of linear equations. As is well known, the system $Ax = b$, where A and b are matrices and x unknown, can be solved by left multiplication of the inverse of A , i.e., $x = A^{-1}b$. However, in MATLAB this is not the best way of solving the system of equations.

```
>> x = A\b
```

computes the solution in a faster and numerically more reliable way. If A^{-1} exists, the solution will be the same as above. Otherwise (that is, for over- or underdetermined systems), the system is solved in least-squares sense.

Solve the system of equations

$$\begin{aligned} 5x_1 + 2x_2 &= 3 \\ 3x_1 + 2x_2 &= 5 \end{aligned}$$

Solution:

More matrix commands

You should try these commands too, as they can become useful later. The **Data processing** commands are particularly important. Note that several of the commands can take several inputs and return more than one output.

Special matrices: zeros, ones, rand, randn, eye, diag.

Processing data: sort, max, min, find, sum.

7 Writing your own functions

Apart from saving scripts in m-files, that could be used to repeatedly invoke a command sequence, it is also possible to write your own functions in MATLAB. A function may take one or several arguments (inputs) and can return one or several return values.

Functions always begin with a line of the form

```
function out = filename(in1,in2,in3)
```

out specifies which variable is going to be returned from the function. filename is the name of the function. The function should be saved in a file named filename.m. in1, in2 and in3 are the arguments of the function. A function can have any number of arguments and return values. If two return values are to be given, the function header takes the form

```
function [out1,out2] = filename(in1,in2,in3)
```

Example 1. Let us rewrite our script (that computed the average grade for two students) as a function:

```
function mean = averagegrade(stud1,stud2)
% Function computing average grade for two students.
% Arguments: Grades for student 1 and 2.
% Return values: Average grade.

disp('The script is running!');
```

```
mean = (stud1 + stud2)/2; % The average grade is computed
disp('Result:');
display(mean);
```

Save the new function as averagegrade.m.

Comparing the function averagegrade with the corresponding script, we can see that the grades of the students were entered directly in the script. Here we let the grades be arguments instead, which means that we let the person running the function decide for what grades he or she would like to compute the average. For the grades 3 and 5 we can write

```
>> m = averagegrade(3,5)
```

There are two main differences between functions and scripts:

1. Functions may have arguments and return values.
2. A script shares the variables with the Workspace from where it was invoked. Functions, on the other hand, create their own workspaces, where only the arguments are defined to begin with. New variables can then be defined inside the function. When all commands of the function have been evaluated, the return value is sent to MATLAB's Workspace, and the workspace of the function disappears (together with its variables).

Many (but not all) of the built-in functions in MATLAB are written in the same way as in the example above. For instance, to see how the function mean is implemented, we can write

```
>> type mean
```

Example 2. Pelle has obtained a loan to buy something nice and expensive (a car, a home home entertainment system, or a sailing boat). He intend to pay 10% of his salary every month until he has payed his debt. The loan has a rate of interest of 1% per month. Write a function where Pelle can enter the size of the loan and his salary, and which computes how large his debt is next month. One suggestion for a solution could be:

```
function new_debt = loancomp(debt,salary)
% new_debt = loancomp(debt,salary)
% Computes the debt after this month.
% Arguments: debt and salary

rate = 0.01;           % Rate of interest per month
paymentsize = 0.1;    % Percentage of salary to pay

new_debt = debt + debt*rate - paymentsize*salary;
```

Write the code in a file named `loancomp.m`. (This file will be used during next exercise session as well.) The function is invoked with

```
>> loancomp(pelles_debt,pelles_salary)
```

where `pelles_debt` and `pelles_salary` are numerical values of Pelle's debt and salary, or variables containing these values.

Compute the size of Pelle's debt next month, if the loan is 10 000 kr, and Pelle earns 25 000 kr/month.

.....

8 Graphics

A picture is worth a thousand words (or so the saying goes). What is drawn by the following code?

```
>> t = 0:0.01:6*pi;
>> y = cos(t);
>> plot(t,y)
```

.....

What is the difference compared to

```
>> plot(y)
```

What goes wrong here?

```
>> t = 0:1:10;
>> y = cos(t);
>> plot(t,y)
```

.....

To plot several functions in the same figure, you can for instance use

```
>> plot(t,y,t,sin(t))
```

Another option is to use the command `hold`. Find out how it works, and plot `cos` and `sin` in the same figure. Your code:

.....

.....

.....
.....

Exercise: The following exercise is adapted from a first course in calculus:

Plot the curve $f(x) = \frac{x^3}{x^2 - 2|x-2|}$. Find all extremum points and the maximum and minimum values, if they exist.

Write a MATLAB function that solves the problem numerically. You have to generate a vector x over a suitable interval with sufficient precision, evaluate the function in these points, and then use a suitable command to find the extremal values. If you feel that it is easier, you can start by writing a script, and then modify it so that you get a function. Name your function `curveplot`. Things to consider:

- Which x values are interesting? How densely should they be sampled?
- You can use `zoom` to magnify parts of the image (or use the magnifying glass icon in the figure window). Another useful command to specify what to show is `axis`. `axis equal` will give the axis the same scale, so a circle will look like a circle.
- The `grid` command adds a grid to the figure.
- What are the pros and cons with this method of studying a function, compared to a traditional analysis (finding the zeros of the derivative etc.)?

Add a title and axis labels to the figure by using `title`, `xlabel`, and `ylabel`. You can print your figure with the command `print`.

Appropriate input could be the ends of the interval for which the function should be plotted. The maximum and minimum function values for that interval should be returned. The first line of the function could be:

```
function [fmin,fmax] = curveplot(xmin,xmax)
```

The function can then be invoked by

```
>> [fmin,fmax] = curveplot(0,10)
```

(The arguments — the values of `xmin` and `xmax` — can of course be changed depending on what interval to investigate.)

To include a MATLAB plot in a report (or similar) might be very illustrative. The figures can be saved using `Save As` in the File menu, or by giving extra flags to the `print` command. Depending on in which context the figure will be used, there are different file formats to choose between. For instance, the `jpg` format might be suitable for publishing figures on the Internet, while `eps` is useful for \LaTeX , and `meta` for files to be included in Word (only in Windows).

9 Homework exercises

General

1. Complete what you did not finish during the computer session!

Matlab as a calculator

1. Apart from `round`, `floor` and `ceil`, there is yet another round off command: `fix`. What is the difference between `floor` and `fix`? Hint: Compare the help of the commands! How are negative numbers rounded off?
.....
2. Use `help log` or some other help function to find out how to compute the following:
 $\log(34) = \dots\dots\dots \log_2(8) = \dots\dots\dots$

- Use `format long` to answer the question: Is $\log(203.8)/\sqrt{9\pi}$ greater than 1?
Are there other ways to find out?

Documentation

- Sometimes you might want to save some of your variables in a file (e.g., when you have performed a long, complex computation, and would like to remember the result for another day). You can do this using the `save` command. Use `load` to make the variables available again. The variables are saved in MATLAB's own format² (MAT files), but you can also save them as text. The command

```
>> save min_fil a b c
```

saves the variables `a`, `b` and `c` in a MAT-file. Try it out!

10 Going further

Investigate two or more of the following topics. Use [1] or MATLAB's own documentation.

More on MATLAB as a calculator and matrices

Suggestions for exercises in [1]: 1, 3, 5, 7, 9, 15, 17, 34

See also the help section "Matrices and Linear Algebra".

²The format has changed between versions 5, 6 and 7 of MATLAB, so if you would like to load variables in another version than they are saved in, please read the documentation for `save`.

More on graphics

More plotting commands: `bar`, `pie`, `stem`, `hist`.

Several plots: `subplot`, `figure`

Documenting figures: `text`, `legend`

You can also investigate how to vary colors and plotting symbols.

Suggestions for exercises: 29, 32

Some commands for 3D graphics: `mesh`, `surf`, `contour`, `meshgrid`, `colormap`

Suggestions for exercises: 39, 41

The MATLAB documentation has a section about graphics, including several demos.

Symbolic computations

Symbolic toolbox can be used to make symbolic computations in MATLAB. See p. 92 in [1].

In the MATLAB documentation, a section on Symbolic Math Toolbox (including demos) can be found.

Numerics and precision

MATLAB as a numerical tool in general is reliable, but in some situations one should be careful. Read more in Chapter 20 of [1].

References

- [1] F. Gustafsson and N. Bergman, *Matlab for Engineers Explained*, Springer Verlag, 2003.