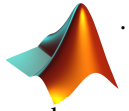
TSRT04: Introduction in MATLAB Computer Exercise/Lesson 1

Version: January 7, 2016

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 and Internet to find out more about MATLAB.

Note: The purpose of this computer exercise/lesson is that you should practice to learn the basics of MATLAB. There are plenty of questions in this document and we suggest that you write down the answers and your own reflections, so it is easy for you to go back. However, you don't have the present the results to anyone. The teachers are here to answer questions and give you feedback on your code. So don't be afraid — there are no stupid questions!

To log on to the computers at ISY you need a student account and a password. Contact support (`support@isy.liu.se`) if you need a new account. Having logged in, you can start MATLAB[1] through the icon  . A MATLAB window will open. It contains several parts, which can be organized differently depending on user settings. The following are the three main 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.

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

2 Preparatory exercises

Before the first lesson you should do the following:

1. Attend the first lecture or read the lecture slides carefully.
2. Watch the MATLAB video tutorials:
   `http://youtu.be/tqjZ80PwqBU?list=PL7CAABC40B2825C8B`

   This contains 8 short videos (45 minutes in total).
3. Start working with Sections 3–4 in this document. 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 documentation is reached by entering

```
>> doc
```

in the command window. Try this yourself and see what happens — every time this document suggests a command, you should try it yourself!

If you want a MATLAB documentation with a more graphical interface, it can also be found online:

`http://www.mathworks.se/help/matlab/index.html`

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: `help help`

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 and laborations:

.....

.....

.....

.....

.....

4 MATLAB as a pocket calculator

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

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

Investigate the difference between the following rows:

```
>> 14/7*2
>> 14/(7*2)
```

Why are the results different?

.....

What (and why?) is the result of the command

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

.....

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

.....

.....

.....

What is `ans`? .....

The up and down arrows of the keyboard can be used to browse the commands you have entered previously, to rerun them and possibly make small changes. 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.

You can also look for “Command History” in the MATLAB window. It lists all your latest commands. You can drag one or multiple rows into the Command Window, you can rerun these rows and make changes to the rows.

Elementary functions

Compute

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

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

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

and how should it be interpreted?

.....

.....

Other related commands

Exponential functions can be entered by using ^ :

```
>> 2^4
```

To use the natural number *e* as basis, there is a special command `exp`:

```
>> exp(4)
```

Other useful elementary functions are the square root, `sqrt`, and different logarithms `log`, `log2`, `log10`:

```
>> sqrt(4)
>> log10(1000)
```

Complex numbers can be created by using `1i` or `1j` as the imaginary unit. Some commands that are relevant to complex numbers are `imag`, `real`, `abs`, and `angle`. Check their meaning!

Display format

Normally, MATLAB displays floating point numbers with 4 decimal digits, but the variables are stored with much higher precision. 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`. It can be used as `clear a` to remove the variable `a`, or simply as `clear` to remove all variables in Workspace. 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, but it less convenient since MATLAB's editor has a "run" button, can help you debug your code, and recommend improvements in the code.)

Write some code in your m-file:

```
disp('The script is running!');
student1 = 3;                  % Grades for student 1 and 2
student2 = 5;
meanvalue = (student1 + student2)/2;  % The average grade
disp('Result:');
display(meanvalue);
```

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. You can also change folder in the top left corner of the MATLAB window.

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.

Look for the "run" button in MATLAB's editor. Press it and see what happens.

6 Matrices

Matrices is the basic data type in MATLAB. In fact, all scalar numbers are simply 1×1 matrices. There are very powerful built-in functions for matrix manipulations and computations in MATLAB.

Construct some matrices:

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

What is the meaning of the semicolons here? .....

Investigate the dimensions of these matrices using the functions `size` and `length`. What are the difference between these two functions?

.....

.....

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 *both* the transpose and the complex conjugate of a 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)` ? .....
`>> G = zeros(1,4)` ? .....
`>> 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?

```
>> index = [1 4 5]
>> y(index)
```

```
>> A(2,2:3)
```

.....

.....

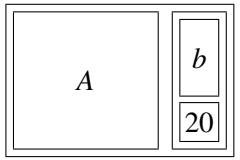
Indexing submatrices can be used to assign to values to parts of a matrix:

```
>> 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 row 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 multiplication `*`, division `/`, and exponent `^`) 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 $\sin^2(x) + \cos^2(x) = 1$ (which holds for any x) by componentwise operations for different x-values; for example, $x = 0.5, 1, 1.5, 2, 2.5, 3, 3.5$ and 4. What will your code look like?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

A common application for matrices is to represent systems of linear equations. These are generally written as $Ax = b$, where A is a known matrix and b is a known column vector. The unknown vector x is then obtained by left multiplication of the inverse of A, i.e., $x = A^{-1}b$. In MATLAB this can be implemented as

```
>> x = inv(A)*b
```

However, this is not the best way of solving linear system of equations in MATLAB, because the inverse `inv(A)` will first be computed separately and then multiplied with `b`. The unavoidable numerical errors that appear when computing `inv(A)` might affect the solution. It also takes unnecessarily much time to compute `inv(A)` when `A` is a large matrix.

In basic courses in linear algebra, we usually solve $Ax = b$ by Gaussian elimination without ever computing the inverse explicitly. We can tell MATLAB to do the same by typing

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

This is a faster and numerically more reliable way of solving the system of equations.

It also worth noting that the computation $x = A^{-1}b$ requires that A is a square matrix and invertible. Otherwise (that is, for over- or under-determined systems), the system can be solved in least-squares sense. This is automatically

done by typing `x = A\b`, while `x = inv(A)*b` will not even work in this case.

Solve the following system of equations using MATLAB:

$$5x_1 + 2x_2 = 3$$
$$3x_1 + 2x_2 = 5$$

Solution: .....

.....

More matrix commands

You should also try the following commands, since they can become useful later. The data processing commands are particularly important. Note that several of these commands can take several inputs and return more than one output.

Special matrices: `rand`, `randn`, `diag`, `linspace`, `inf`.
Data processing: `sort`, `max`, `min`, `find`, `sum`, `prod`.

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 (outputs).

Functions always begin with a line of the form

```
function output = filename(input1,input2,input3)
```

`output` specifies the name of the variable that will be returned by the function. `filename` is the name of the function. The function should be saved in a file named `filename.m`. `input1`, `input2` and `input3` are the arguments of the function. This is the information that the user needs to give to the function to enable its computations. 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 [output1,output2] = filename(input1,input2,input3)
```

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

```
function meanvalue = averagegrade(student1,student2)
% Function computing average grade for two students.
% Arguments: Grades for student 1 and 2.
% Return values: Average grade.

disp('The function is running!');
meanvalue = (student1 + student2)/2;  % The average grade
disp('Result:');
display(meanvalue);
```

Save the new function as `averagegrade.m`.

Comparing the function `averagegrade` with the corresponding script in Section 5, 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
```

How can we use `mean` to do the same computation as `averagegrade`?

.....

.....

Exercise 2. Emma has obtained a loan to buy something nice and expensive (a car, a home entertainment system, or a sailing boat). She intends to pay 10% of her salary every month until she has paid her debt. The loan has a rate of interest of 1% per month. Write a function where Emma can enter the size of the loan and her salary, and which computes how large her 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
interest = 0.01;          % Rate of interest per month
paymentsize = 0.1;        % Percentage of salary to pay

new_debt = debt + debt*interest - 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(emmas_debt,emmas_salary)
```

where `emmas_debt` and `emmas_salary` are numerical values of Emma's debt and salary, or variables containing these values.

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

.....

8 Visualization

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)
```

Why does the following figure look so crappy and how can make it look more like cosine?

```
>> 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 3. The following exercise is adapted from a first course in calculus.

Plot the curve $\frac{x^3}{x^2-2|x-2|}$ for a suitable range of x-values. Find all extremum points and the maximum and minimum values, if they exist. You can use `max` and `min` to do this.

Write a MATLAB function that solves this problem numerically. You have to generate a vector `x` over a *suitable* interval with *sufficient* precision, evaluate the function in these points, plot the function, and then use a suitable command to find the extremal values.

It is recommended that you first write the code as a script (so you can easily access variables in the workspace for debugging) and then modify it so that you get a function. Name your function `curveplot`.

Things to consider:

- Which x values are of interest? 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 shown is `axis`. In addition to zooming, `axis equal` will give the axis the same scale, so a circle will look like a circle and not an ellipse.
- 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`.

Appropriate input to your `curveplot` function could be the ends of the interval for which the function should be plotted. The maximum and minimum

function values for that interval should also 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, and might be the main reason that you use MATLAB in later courses.

The figures can be saved using Save As in the File menu. 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 or to use it in MS Word, while EPS and PDF are useful for LaTeX. MATLAB has also its own FIG format which allows you to open the figure in MATLAB again. It is recommended to always save in FIG format, in addition to other formats, so that you can later make changes in the figure (change the zoom, add more curves, etc.)

9 Homework exercises: Week 1-2

General

1. Install MATLAB on your own computer (if you have one)
2. Complete what you did not finish in this document during the first computer session!
3. Some additional homework exercises are listen below. We recommend you to solve these during the first weeks of the course.

MATLAB as a pocket 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 handled?

   .....

2. Use `help log` or some other help function to find out how to compute the following:
   $\log(34) =$ ..... $\log_2(8) =$ .....

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

4. How many decimals of π can you find by changing the format?

   .....

5. What does the command `eps` do? Compute `eps^2`, `1+eps`, and `1+eps^2`. Explain the unexpected behavior.

   .....

   .....

Documentation

1. 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 format (.MAT files), but you can also save them as text. The command

   ```
   >> save my_file a b c
   ```

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

   Empty the workspace and use `load` to retrieve the variables from your .MAT file.

Matrices

1. Construct a diagonal matrix where the main diagonal contains the values 1 1 2 3 5 8 13 and all other elements are zero. Which function can you use?

   .....

2. Construct the following matrix using `reshape` and/or `repmat`:

$$\begin{bmatrix} +1 & +1 & +1 \\ -1 & -1 & -1 \\ +1 & +1 & +1 \\ -1 & -1 & -1 \end{bmatrix}$$

   .....

   .....

3. Verify that the determinant satisfies $\det(I+AB) = \det(I+BA)$ for any compatible matrices A and B. These matrices need not be square matrices. You can for example let A be 4×3 and let B have dimensions 3×4. You can generate matrices randomly with any dimensions using the commands `rand` and `randn`.

Visualization

1. Verify that $\sin(t) = \frac{e^{it}-e^{-it}}{2i}$, for all t, by plotting $\sin(t)$ and $\frac{e^{it}-e^{-it}}{2i}$ separately in a figure. Notice that i is the imaginary number. Use different colors to separate the curves.

2. Create 10000 random elements (e.g., a 10000×1 matrix) using `rand` or `randn`. Study the distribution of the random numbers by making a histogram with `hist`. To get a decent resolution you can set the number of bins to 50. What is the difference between the randomness in `rand` and `randn`?

   .....

   .....

3. Make a 3D visualization of $\text{sinc}(x^2+y^2)$ for $-10 \leq x \leq 10$ and $-10 \leq y \leq 10$. You can, for example, use the functions `sinc` and `meshgrid` to evaluate the function. Plot the results using `mesh` and `surf`. What is the difference between these commands? How can you choose the colors of the graph?

   .....

   .....