Introduction to MATLAB

I. General Remarks

MATLAB means **MAT**rix **LAB**oratory and is a programming language/Integrated Development Environment for technical and scientific applications.
The strengths of the programming language are vector/matrix calculus, simulations and scientific visualization.

Remarcks concerning the syntax:

- Keep in mind: sensitive to lower- and uppercase letters!
- all MATLAB commands are written using lowercase letters
- interactive input is accepted in the Command Window after the prompt **>>**
- Press **<Return>** to execute the command
- To exit MATLAB type **quit + <Return>**

II. Simple Calculations with scalars

The file

scalars+operations.txt

contains simple MATLAB commands. Execute these commands line by line (one after the other). You can just copy the lines from the file to the command window!

Note: Try to understand the output!

You will recognize the following details:

1. Basics

- operations for **basic artithmetics** are **+, -, *, /**
- the symbol **^** means **power of**

2. Backslash operator: **a\b** is equivalent to the expression **inv(a)*b**; **inv(a)** calculates the inverse (here: reciprocal of a).

3. With scalars this is simple and straight forward. However, it gets **important** as soon as we use **matrices** ! Why?

4. MATLAB knows **pi**.

5. Imaginary units are written with **i**.

6. The constant **eps** gives the (relative) machine precision.

7. MATLAB provides the following functions

-	**abs**	absolute value
-	**sqrt**	square root
-	**exp**	exponential function
-	**log**	natural logarithm

.... sin, cos, tan, asin, acos, atan and many others ....

8. The assignment operator **=** is used to assign any value to a variable.

9. The **clear** command deletes an assignment.

10. By default MATLAB prints **five decimal places**, though the precision used internally is much higher!

11. Use **format long** to get more decimal places (undo: **format short**).

12. Terminating a line with a semicolon prevents any output; the command gets still executed.

III. Vectors and Matrices

Now execute the commands in the file

matrices+calculation.txt

The following points will become evident:

1. Vectors are matrices with one column (1 x n) or one row (n x 1).

2. Matrices can be specified explicitly using **brackets** [ ]; columns are separated by commas **,** or blanks, rows are separated by semicolons **;.**

3. A tick □ will **transponse** a matrix.

4. The dimension of (row- or column-) vectors is the return value of **length**, the dimension of a matrix is returned by **size**.

5. You can add and subtract matrices of **similar size**.

6. Multiplication **by a scalar value** requires ***.** *(the dot is important!!)*

7. Matrix multiplication ***** requires two matrices with **compatible** column and row **dimension**. Otherwise you will get an error message! Remember □

(m x n) * (n x k) = ? x ?

8. The power of a square matrix can be calculated using **^**.

9. Using the operations **.*, ./** and **.^** with matrices of **similar size** means element-wise (element-by element) multiplication, division und power. It is allowed to use scalars with these operations.

10. Elementary functions e.g. **abs, sqrt, sin** etc. can be used with matrices and vectors. Evaluation is done element-by-element.

11. The functions **zeros** (null matrix), **eye** (identity matrix)
and **ones** (matrix with only ones) generate special matrices.

The functions **rand** and **randn** generate random matrices with elements drawn from an uniform and normal distribution respectively.

The expressions **from:to** and **from:stop:to** are used to produce arithmetic series. These kind of expressions are important for indexing and loops!

Elements of a matrix A can be selected using **A(row,col);** it's also possible to use an index range **from:to.**

Using just the **colon :** without specifying a range addresses the entire row/column. Note: partial matrices can be used on the left hand side of an assignment!

diag return the diagonal of a matrix or returns a diagonal matrix .

inv calculates the inverse of a square matrix, **det** calculates the determinant.