

Short Introduction to MATLAB

MATLAB is a strongly uprising simulation and calculation tool for all kinds of applications. Its success is based on its easy use and its universal structure: Variables do not need to be defined and a wide variety of commands and tools boxes are available. MATLAB is not a Computer language but an interpreter, i.e. the commands refer to subroutines, mostly written in C-code.

Sometimes, the huge variety seems to confuse the user. However, once the basics are well known, it is easy to get along with MATLAB very well.

MATLAB is typically used for research and developing applications, because difficult equations can be programmed in just a few lines. On the other hand, MATLAB requires rather long calculation times, and therefore it is not used for mass applications. In our case, we do not count for a short calculation time but we rather profit from the fast program generation and the variety of already defined commands. Besides, MATLAB is a vector oriented program. This means that vectors can be used like on paper with many kinds of vector operations.

Getting Started

In Windows, MATLAB is started out of the menu, in Unix system, type “matlab” in the shell. Then, the command window will show up.

In the command window, you can type all commands, for example

```
>> a=3+5 and <return>
```

will result in

```
a =
```

```
8.
```

This is a useful property of MATLAB, it serves like a calculator.

For us, it is even more important to write a MATLAB code like in a programming language. Therefore you click on FILE/NEW (Windows) or you create a name.m-file using an editor, for example emacs (Unix).

This m-file now can contain as many commands as you wish, and all of them are performed by MATLAB. Make sure that you are operating MATLAB in the same folder where your m-file is located. In the command window, you can use shell commands to operate (cd, pwd, dir ...).

If you type the name of your mFile in the command window, the code will be executed, as if you would type the commands one after each other in the command window.

Notice that commands can be copied and pasted while still be executed!

Important commands

MATLAB offers a very good online-help including many examples and applications. Once you know the fundamentals, you will find a systematic way to teach yourself the commands. In this introduction, some of the commands for the homework project are presented, far from completeness. So you might use other commands as well.

“Clear”: This clears all the variables. Otherwise, all variables are still available in the memory.

`a=1;` %If you type a semi-colon behind the command, the result is not shown on the screen, otherwise it will be presented:

```
>> a=1 <enter>
```

```
a =
```

```
1
```

%: Comments, line is not executed from that point on

```
>> column = [1; 3 ;4]    %column vector
```

```
column =
```

```
1
```

```
3
```

```
4
```

```
>> row = [1 3 4]        % row vector
```

```
row =
```

```
1 3 4
```

```
>> product = row*column % vector product
```

```
product =
```

```
26
```

```
>> row = row'    % transpose of vectors and matrices: ‘
```

```
row =
```

```
1
```

```
3
```

```
4
```

*>> product = row.*column %now, only the corresponding components are multiplied.*

product =

*1
9
16*

For all vector operations, watch out for the right dimensions, as you would do it on paper!

>> A=[2,1;1,3] %matrix

A =

*2 1
1 3*

>> f=[4 ;7] %column vector

f =

*4
7*

>> x=A\f %solves A x = f

x =

*1.0000
2.0000*

Functions are a good tool for vectorized use.

The function

```
function y = f_1d(x)  
y=log(x);
```

is saved in the same folder as f_1d.m.

Now, create in the command window a vector

>> x=(1:.2:2) <enter>

x =

1.0000 1.2000 1.4000 1.6000 1.8000 2.0000

and then

```
>> y=f_1d(x) <enter>
```

```
y =
```

```
0 0.1823 0.3365 0.4700 0.5878 0.6931
```

gives $y(x)$. You could also type directly $y = \log(x)$, but you can use this scheme to write your own functions.

>> plot(x,y) will give a plot of $y(x)$.

A short example for conditions (if) and loops:

```
for i=1:1:3    %start number : step size : end value , all of them have to be integer
    if i>=2
        a(i)=1;
    else
        a(i)=2;
    end
end
end
```

results in

```
a =
```

```
2 1 1.
```