Matlab Basics
Lecture 1

Juha Kuortti, Heikki Apiola

October 22, 2018

Meet the IDE

Getting Help

Basic scalar variables.

1. What, where, how

- **Mat**rix **lab**oratory [Cleve Moler, Mathworks inc.]
- Language and tool for numerical computation
- Large number of mathematical and other functions.
- Functional programming language, user can extend Matlab by defining (programming) own functions.
- Application specific toolboxes
- http://se.mathworks.com/help/matlab/index.html
- http://www.mathworks.se/academia/
- http://se.mathworks.com/help/matlab/examples/basic-matrix-operations.html?prodcode=ML
- google: learn matlab, matlab <keyword>

help,doc,lookfor

- help, doc
  - `>> doc` starts help system, same as $\boxed{?}$
  - `>> help name` `>> doc name`

    help is faster, **doc** is more comprehensive.
  - Some search words for help/doc:

    elfun – elementary functions

    general, ops, elmat, ... More on next slide
- **lookfor**

  ```
  >> lookfor sum, lookfor solve
  >> lookfor optimize, lookfor equation
  ```

  Beware: Some searches may give too many hits.
- google Matlab,<keywords, phrases>

Some help-keywords »help

```
general         - General purpose commands
ops             - Operators and spec. chars
lang            - Programming language constructs
elmat           - Elementary matrices
elfun           - Elementary functions
specfun         - Special functuons
matfun          - Matrix functions
datafun         - Data analysis and Fourier transform
graph2d         - 2d graphics
graph3d         - 3d graphics
graphics        - Handle graphics
imagesci        - Image and scientific data
demos           - Examples and demo's
```

First steps and concepts

- Workspace, command window
  - Matrices and other datatypes are stored in memory, contents are shown in **workspace.**.
  - » `who, whos`

- Commands (functions) are applied to variables in the workspace.
  - Matlab *interprets* and returns the result(s) in the workspace. (Or displays an error

First steps and concepts

- Workspace, command window
  - Matrices and other datatypes are stored in memory, contents are shown in **workspace.**.
  - » `who, whos`
- Commands (functions) are applied to variables in the workspace.
  - Matlab *interprets* and returns the result(s) in the workspace. (Or displays an error

1. Start Matlab
2. Create a working directory: Either File-menu or command `>> mkdir mydir` [a])
3. `>> cd mydir`
4. Create variable: `>> x=5`
5. Do: `>> y=exp(x)`
6. Try `>> who, whos`

[a] Some Unix/Linux-commands can be given in the Matlab command window

Working in the command window

- "Undoc" command window (or make it large enough)
- Here's a possible first session, try yourself!

```
>> 3/4
ans =
    0.7500
>> 4*ans
ans =
     3
>> r=3/4;  % Supress output
>> r       % Show result
r =
    0.7500
>> Area=pi*r^2
Area =
```

Arithmetic operations, examples

- Multiplication and division from left to right, equal precedence.
- Ordinary precedence rules. **Use parentheses** for clearity !

```
>> 6/3*2
   ans = 4
>> 6/(3*2)
   ans = 1
```

```
>> 6/3/2
   ans = 1
>> 6/(3/2)
   ans = 4
```

Exercise

Make the following variables:

- a $= 10$
- b $= 2.510^{23}$
- c $= 2 + 3i$ (i being the imaginary unit)
- d $= e^{\frac{2}{3}\pi i}$

Arithmetic, precedence

Scalar arithmetic operations

Symbol	Name	Math	Matlab
`+,-`	add/subtract	$a \pm b$	`a+b,a-b`
`*`	multiply	ab	`a*b`
`/`	Righ divide	$\frac{a}{b}$	`a/b` [1]
`\`	Left divide	$\frac{b}{a}$	`a\b` [2]
`^`	power	a^b	`a^b`

[1] Recommendation:Use this for scalar division

[2] Recommendation:Use this for "matrix division"

Command window, history, create script

Command window:

- Use the up-arrow key to scroll back through the commands.
- Use the down-arrow key to scroll forward
- Edit a line using the left- and right-arrow keys.
- Press the Enter key to execute the command

Create script from command history:

- Choose commands from the history with CTR + mouse left. Mouse right lets you choose "create script". (More on scripts soon.)
- Execute commands from the editor: CTR-Enter.

Little scalar task, work together

- The volume of a circular cylinder of height h and radius r is given by $V = \pi r^2 h$. A particular cylindrical tank is 15 m high and has a radius of 8 m. We want to construct another cylindrical tank with a volume 20 percent greater but having the same height. How large must its radius be?

Solution, command history, make script

Here's the Matlab-session:

```
>>r = 8;
>>h = 15;
>>V = pi*r^2*h;
>>V = 1.2*V;          % 20% increase in V
>>r = sqrt(V/(pi*h))
r =
8.7636
```

Use $\boxed{\uparrow}$ for command history. With CTR+Mouse left paint commands you want to save, press mouse right and choose "make script".

Scripts, publish

You can perform operations in MATLAB in two ways:

- In the interactive mode, in which all commands are entered directly in the Command window.
- By running a MATLAB program stored in a script file. This type of file contains MATLAB commands, so running it is equivalent to typing all the commands—one at a time—at the Command window prompt. You can run the file by typing its name at the Command window prompt.
- The script file commands can also be executed directly from Matlab's editor window either by parts or all of them.
- `publish` produces a well structured document of running the script.

Examples of expressions

```
>> 6*sqrt(2)+pi^2
   ans=18.3549
>> one=sin(pi/3)^2 + cos(pi/3)^2
  one =  1
>> 1==sin(pi/3)^2 + cos(pi/3)^2    %  Equal?
  ans = 1                         %  Logical: true
>> exp(i*pi)                      %  Not e^x !!
>> 1.0/0.0 ->  Inf
>> -4/Inf  ->  0
>> 0/0     ->  NaN  % "Not-a-number".
>> format long      % Show max number of digits.
>> [1+eps,1+3*eps]  % eps: Limit of rel. accuracy.
>> format short     % Back to default display.
>> clc              % Clean display
```

Workspace

- Variables are stored in the memory and accessed in the workspace
- Commands for managing the workspace are called here "system commands", perhaps a little "unofficially". For instance who, whos show variables in the workspace, latter with sizes.
- clear erases all variables from the workspace (memory), clear var1 var2 erases these variables.
- The syntax of "system commands" differs from computational and other functions. System commands don't use parenteheses or commas.

Some “system commands”

Some commands for managing the workspace

Matlab command	Description
`clc`	Clear command window (visually).
`clear`	Clear all variables (from memory).
`clear var1 var2`	Clear these variables.
`who`	List variables in memory
`whos`	List variables with sizes in memory
`format`	Display format of numbers
`clf`	Clear current graphics window.
`close all`	Close all graphics windows.
`shg`	Show Graphics.

Comparison, relations, scalar case

- Remember: `name = expression` means assignment of the value of expression to variable `name`.
- `lhs == rhs` Returns 1 if equal,0 if not.
- `<, <=, >, >=`, $\sim=$ are other arithmetic comparisons.
- The value of a comparison is true (1) or false (0).
- Precedence of arithmetics is higher than that of comparisons

```
>> 1==0                    % --> ans = 0
>> E = 1.733>tan(pi/3)     % --> E = 1
```

What are the results ? : `>> E=4>5-2 , (4>5)-2`

Expression, variable, special variable *ans*

- An expression consists of numbers, variables, functions, operators such as
  `+,-,*,/,^,(), sin, cos, exp, abs, ...`
- help/doc ops,elfun [See previous slide for more searchwords.]
- `>> var=expression`
  assigns the value of *expression* to variable *var*.
- If the expression is written without an assignment, the result is assigned to the special variable ans.
  Note: ans holds just the previous result, the next such computation overwrites it.

Variable names and types

Variable names:

- Start with a letter, then letters, numbers, underscore(_)
- Other special characters not allowed, especially minus (-) is not possible, as it means subtraction.
- CASE SENSITIVE! (var1 is different from Var1)

NOTE: Matlab help texts: old style (from 1980's) of capitalized NAME meaning **name**, Let's abandon this usage.

```
>> number=-2.345
>>        % Note: period (.), not comma (,)
>> complex_number=3+4*i
>> n=1;n=n+1;
```

Variable names and types

- No need to initialize or define a variable, if efficiency is not an issue (return to this later).
- Default type is 64 bits floating point number ("double"), about 16 decimal digits.

```
>> 2.345
```

- Characters are of type 'char' (16 bits)

```
>> 'this is a character string'
```

- Change numeric data into character

```
>> num2str(2.3)
>> str2num(ans)    % and back
```

- Other tyes: `logical, single,int-types`
  `help datatypes`
  https://se.mathworks.com/help/matlab/numeric-types.html

Complex numbers

- All arithmetic in Matlab works on complex numbers as well. Matlab has special variables `i` and `j` for $\sqrt{-1}$.
- All special variables can be overwritten, so:

```
>> 2+3*i
ans =
   2.0000 + 3.0000i
>> i=1;
>> 2+3*i
ans =
     5
>> clear i
>> i
ans =
   0.0000 + 1.0000i
```

Complex numbers continued

```
>> sqrt(-1)
ans =
   0.0000 + 1.0000i
>> 4 + 6*j;
>> 4 + 6j;   % Correct, I don't recommend:
>> 4+j6   % -> Undefined function or variable 'j6'
>> x=1;y=2;x+y*i
>> x+yi;            % Same error.
>> C=1 - 2*i;
>> real(C), imag(C)
>> abs(C)
>> angleDegrees=angle(C)*180/pi
>> exp(i*pi)  % Matlab meets Euler!
ans =
```

Dimensional increase

Vectors

Matrices

Arrays.

Vectors, arrays, matrices

Basic data structure: Matrix (array), elements: complex numbers.
Let's limit ourselves at first to two-dimensional arrays.

```
>> rowvect=[1 2 3 4]  % List of elements
>> 1:4                % Same with colon(:)-operator
 ans = 1  2  3  4
                     % ans:previous non-assigned result
>> colvect=[1;2;3;4]
>> v'                        % Transpose of row-vector
>> length(rowvect)           % Nr. of elements
     ans = 4
```

Vectors, matrices,arrays (continued)

```
% Matrix and its size
>> A=[1 2 3 4 ;5 6 7 8; 9 10 11 12]
>> [m,n]=size(A)
  m=3,n=4
>> [size(A,1) size(A,2)]
ans =
   3     4
>> who, whos     % show workspace variables
```

- Column vector: `(m,1)`-matrix
- Row vector: `(1,n)`-matrix
- Scalar: `(1,1)`-matrix
- Empty: `(m,0)` or `(0,n)` matrix

Calculus with vectors

- The numbers $0, 0.1, 0.2, \ldots, 10$ can be assigned to the variable `u` by typing `u = 0:0.1:10;`
- `length(u)` reveals us that there are 101 elements in `u`.
- To compute $w = 5 \sin u$ for $u = 0, 0.1, 0.2, \ldots, 10$, the session is;

```
>>u = 0:0.1:10;              % By 'misuse' of Matlab:
>>w = 5*sin(u);              >>for k=1:length(u)
                               w(k)=5*sin(u(k));
                               end;toc
```

- This was our first acquaintance with "vectorization".

Vector excercise

Make the following variables:

- aVec $= \begin{bmatrix} 3.14 & 15 & 9 & 26 \end{bmatrix}$
- bVec $= \begin{bmatrix} 2.71 \\ 8 \\ 28 \\ 182 \end{bmatrix}$
- cVec$= [5, 4.8, \ldots, -4.8, -5]$ (all the numbers from 5 to -5 with increments of -0.2)
- dVec $= [10^0\, 10^{0.01}\, \ldots 10^{0.99}\, 10^1]$ logarithmically spaced numbers between 1 and 10.
- eVec = `'Hello there'` (eVec is a string, which is a vector of characters)

“Scalar functions” support vectorization

The previous example leads us to the following general idea: Functions which applied to a scalar produce a scalar result are called *scalar functions* (perhaps a bit misleadingly). When such functions are applied to a vector, they operate on every element of the vector. Mathematical functions `help elfun, specfun` among others are of this type.

```
>> t = [-1 0 1];
>> y = exp(t)
y =
    0.3679    1.0000    2.7183
>> [exp(-1) exp(0) exp(1)]
ans =
```

“Scalar functions” support vectorization (contnd.)

Assume we want to compute values of

$$y = e^{-x} \sin x$$

at a vector x. We need the vector
$y = (e^{-x(1)} \sin(x(1)), e^{-x(2)} \sin(x(2)), \ldots, e^{-x(n)} \sin(x(n)))$
Here we need the pointwise product `(.*)` of two vectors:

```
>> x=-pi:.1:pi;
>> y=exp(-x).*sin(x);
```

This is just the data we need for plotting. `>> plot(x,y)`

Functions for building vectors
colon(:),linspace,logspace

- `v=a:b, w=a:h:b;` default: h=1
- `v=linspace(a,b,N);` default: N=100
- `v=logspace(a,b,N);` $10^a, \ldots, 10^b$, N points

```
>> 0:10; 0:.1:1;
>> 10:-2:0
  ans =
    10     8     6     4     2     0
>> logspace(0,1,4)
   ans =
    1.0000    2.1544    4.6416   10.0000
>> 10.^linspace(0,1,4)
   ans =
    1.0000    2.1544    4.6416   10.0000
```