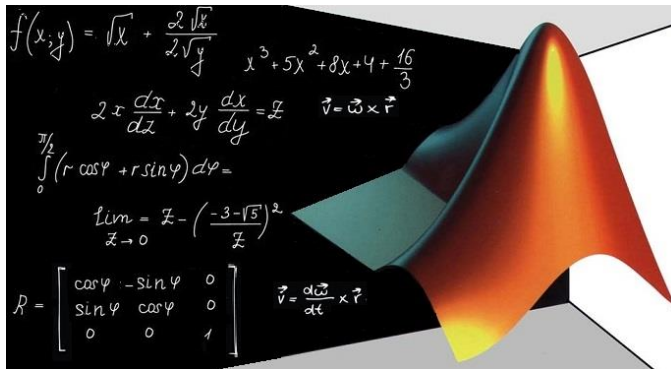


Getting started with Matlab



Andrea Greve

MRC Cognition and Brain Sciences Unit

23.10.2018

MATLAB (MATrix LABoratory)



What is MATLAB

Simple programming language, high-performance

Based on matrix representations

Can process and visualize data

Typical uses

Math, calculation and computation

Data analysis, exploration, and visualisation

Modelling, simulation and experimentation

Why do I need to program?

Provides independence and flexibility to do with your data what you want

Not limited like off-the-shelf software and scripts written by others

Is an efficient and easy way to reproduce analysis (save time)

Some commonly used software in research is based on MATLAB (SPM for fMRI/MEG analysis)

Getting started with MATLAB

Starting matlab

On UNIX: type matlab at command prompt

On a PC: click on the MATLAB icon if you are

Issues on startup

MATLAB needs a connection to the license server

Check internet connection

Too many users can use all available licenses

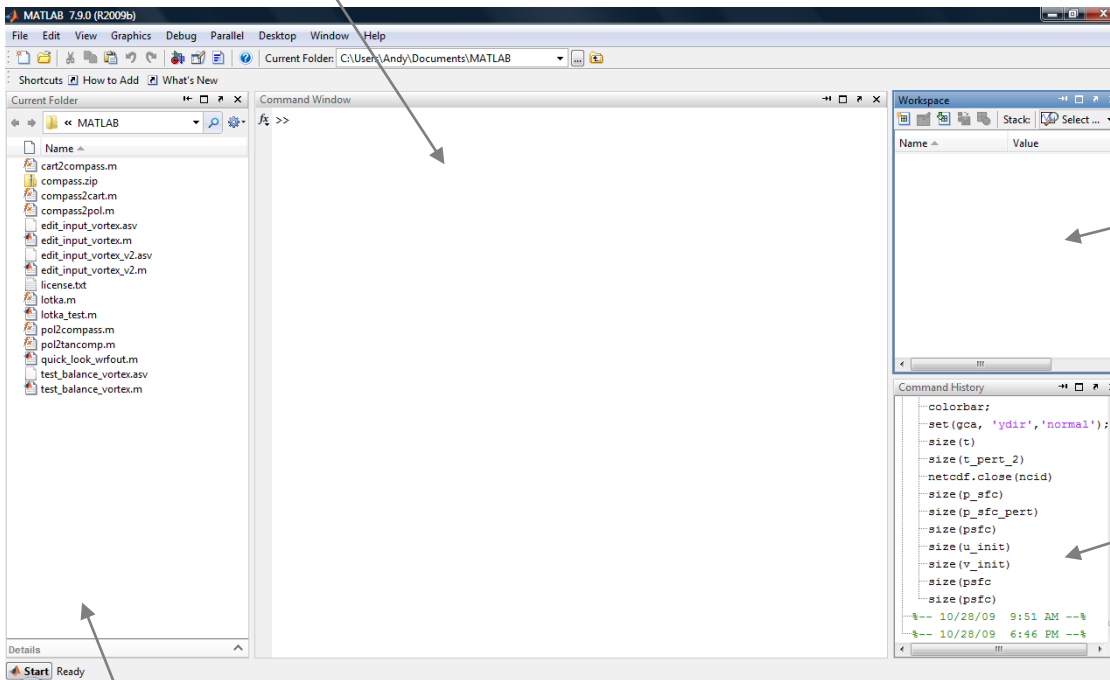
Try again later

MATLAB components: screen

Command Window
type commands

Workspace

View program variables
Double click on a variable
to see it in the Array Editor



Command History

view past commands
save a whole session
using diary

Current Directory
View folders and m-files

MATLAB display

MATLAB displays

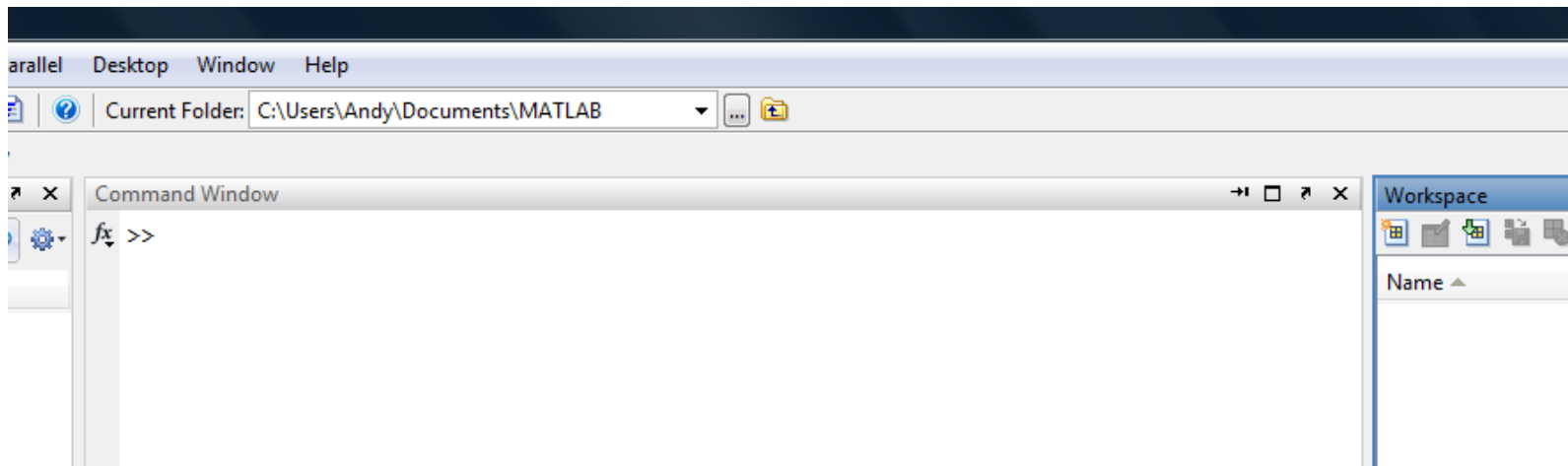
>> prompt when ready for a command

no >> prompt when processing commands

newer versions also say “Ready” or “Busy” in lower left corner of GUI

can use arrow keys to work through command history and modify

Commands are essentially the same as UNIX command prompt



Types and classes

characters	'h' (string, multiple characters e.g. 'hello')
numbers	3, 5.5
vectors	one-dimensional set of elements (of same type)
matrices	multi-dimensional set of elements (of same type)

Matlab is case sensitive: Data, DATA and data (are all different)

Typing your first command into matlab

```
> 2
```

```
ans =
```

```
2
```

```
> a
```

```
??? Undefined function or variable 'a'.
```

```
> 'a'
```

```
ans =
```

```
a
```

```
> a = 2
```

```
a =
```

```
2
```

```
> a2 = 2
```

```
a2 =
```

```
2
```


Typing your first command into matlab

```
> x = 2
```

```
x =
```

```
2
```

```
> y=2
```

```
y =
```

```
2
```

```
> x + y
```

```
ans =
```

```
4
```

Typing your first command into matlab

```
> z = x + y
```

```
z =
```

```
4
```

```
> x = x + x
```

```
x =
```

```
4
```

```
> x = x + x
```

```
x =
```

```
8
```

```
> h = x + x
```

```
h =
```

```
16
```

MATLAB as calculator

Elementary mathematical operations:

>> 2+3 addition '+' / subtraction '-'
ans = 5

>> a=2*3 multiplication '*' / division '/'
a = 6

>> b=(a+2)/4
b=2

Variable 'X' automatically allocated

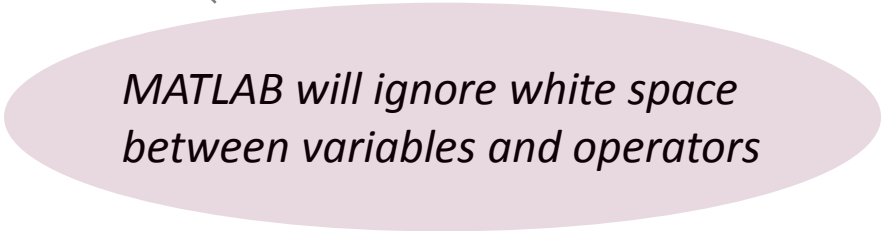
*MATLAB does not require declaration of variables
nice, but can get you in trouble so be careful*

Elementary mathematical operations:

>> c=log(2.7183) natural logarithm 'log' & many other functions
c = 1

MATLAB: arithmetic operators

		Operation	MATLAB form
Exponentiation:	\wedge	a^b	a^b
Multiplication:	$*$	ab	$a*b$
Right Division:	$/$	$a / b = a/b$	a/b
Left Division:	\backslash	$a \backslash b = b/a$	$a \backslash b$
Addition:	$+$	$a + b$	$a+b$
Subtraction:	$-$	$a - b$	$a-b$



*MATLAB will ignore white space
between variables and operators*

Typing your first command into matlab

```
> a > 0
```

```
ans =  
    1
```

```
> x + x > x
```

```
ans =  
    1
```

```
> x < x
```

```
ans =  
    0
```

MATLAB: relational and logical operators

	Operation	MATLAB form
greater than	>	<code>r>0</code>
less than	<	<code>x<1</code>
greater than or equal	>=	<code>r>=x</code>
less than or equal	<=	<code>x <= eps</code>
equal to	==	<code>p == pi</code>
not equal to	~=	<code>p~= pi</code>
and	&	<code>a>3 & c<0</code>
or		<code>g>3 g<0</code>

MATLAB: Syntax

Syntax

Set of rules that define the combination of symbols to become a program within a particular programming language.

It is the *vocabulary and grammar* for writing programming code which can be *unambiguously* understood by the programming language.

```
>> y = (X+2)/4
```

Evaluation occurs from left to right

When in doubt, use parentheses MATLAB will help match parentheses

Typing your first command into matlab

> `[x, 2*x, 3*x, 4*x]` one-dimensional array: vector

ans =

8 16 24 32

> `xyz_vector = [x y z]` use meaningful names

xyz_vector =

8 2 4

> `sort([x y z])` functions

ans =

2 4 8

> `sort(xyz_vector)` functions

ans =

2 4 8

Typing your first command into matlab

```
> xyz_vector(1)
```

```
ans =
```

```
8
```

```
> xyz_vector([2 1])
```

```
ans =
```

```
2 8
```

```
> 1:7
```

```
ans =
```

```
1 2 3 4 5 6 7
```

```
> 1:2:7
```

```
ans =
```

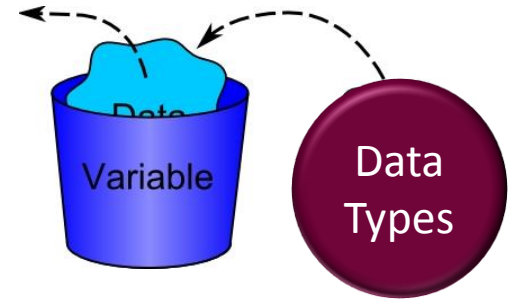
```
1 3 5 7
```

```
> xyz_vector(2:3)
```

```
ans =
```

```
2 4
```

MATLAB: Variables



A variable: a place in memory with a name that contains a value.

Variables types:

Numeric: single element (scalar), array, multi-dimensional array

```
>> X=5
```

Text: character, string (array of characters).

```
>>var_name = 'hello'
```

Variables names:

Matlab is *case-sensitive*

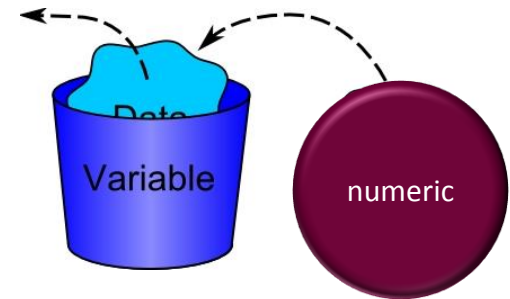
refer to variable in *exactly* the same way: *typos* are unacceptable

Variable names must start with a letter, can be followed by any combination of letters, numbers and underscores (no special characters)

don't use same name as functions: min, max, sqrt, cos, sin, tan, mean, etc

don't use MATLAB reserved words: if, end, date, etc

MATLAB defining numeric variables



`x = 1 ;` (scalar, integer)

`numSubjects = 8;` (meaningful name)

`myScalar = 1.1;` (scalar, rational (decimal) number)

`myVec = [1 2 3];` (one-dimensional array: vector)

`myVec = [1.2 2 3];` (one-dimensional array with mixed integers and rational numbers)

Semicolon (;) at the end of a command prevents echo in the command line

MATLAB: arrays and indexing

Array: a set of ordered elements.

Defining: arrays are created by assignment

```
>>myVec = [3 1 7 9 4];    the index of 7 is 3  
myVec = 3 1 7 9 4
```

Indexing: every element in the array has a place called index.
the i-th element is the element in the i-th place.

Retrieval: getting an element from a specific index in the array.

```
>>myVec(3)                arrayName(index)  
ans = 7
```

Assignment: an element can be replaced:

```
>>myVec(3) = 5            arrayName(index) = newValue  
myVec = 3 1 5 9 4
```

myVec(1)	myVec(2)	myVec(3)	myVec(4)	myVec(5)
3	1	7 -> 5	9	4

Let's try it out

MatlabBasic.m: Example 1-4

write a script

copy and past in command line

highlight, right klick and “evaluate selection”

highlight and press key ‘F9’

run the script

myVec(1)	myVec(2)	myVec(3)	myVec(4)	myVec(5)
3	1	7 -> 5	9	4

```
% variable
myVariable = 9 % with echo to the command line
myVariable = 9; % semicolon at the end suppress echo at the command line

% Example 2 - Arrays
firstArray = [2 8 90 46 -3 5]
emptyArray = []

% Series
simpleSeries = 1 : 10 % default increment is 1
seriesArray = [2 : 4 : 20]
seriesArray2 = [3 : -1 :1]

whos

% Retrieval
firstArray
oneElementRetrieval = firstArray(2)
arrayRetrieval = firstArray([5 3])

% Assignment
firstArray(2) = 100
firstArray([4 1]) = [52 7]
firstArray([2 : 3]) = []

% Check the size/length
size(firstArray)
length(firstArray)

% Replacing elements:
firstArray([1 3]) = firstArray([3 1])

% Example 4 - Special indexing
firstArray(2 : end)
firstArray(end-2:end)
firstArray(:)
```

Practice 1

Open **Matlab** and change the current directory to a folder of your choice.

In the command window, do the following:

Create a 1x5 array with numeric values as you like.

Find the variable in the workspace and double-click it to see its content.

Change the value of the 3rd element in the array. Make sure you can see this change in the workspace.

Delete the 4th element in the array.

Use **'size'** function to check for the size of the array.

Use **'length'** function to check for the length of the array.

Clear all the variables and command window using **'clear'** and **'clc'**.

MATLAB: Matrices

Matrix: 2D array (table).

Elements are ordered in 2 dimensions: rows and columns.

M x N matrix: M rows, N columns.

```
>> myFirstMat = [1 2 3; 4 5 6];
```

```
myFirstMat =
```

```
1 2 3  
4 5 6
```

N=5

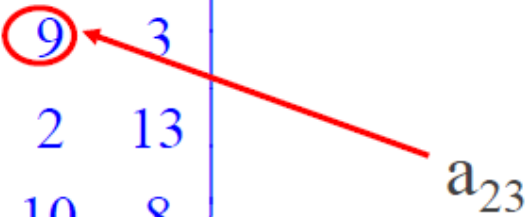
N=4

MATLAB: Matrices indexing

Indexing:

the **aij** element is the element in the **i-th** row and the **j-th** column.

- Example:

$$\begin{pmatrix} 5 & 8 & 12 & 4 \\ 7 & 1 & 9 & 3 \\ 11 & 5 & 2 & 13 \\ 3 & 6 & 10 & 8 \end{pmatrix}$$


a_{23}

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

MATLAB: Arrays

Scalar: 1x1 array.

Vector: one-dimensional array.

Row: 1 x N array.

Column: N x 1 array

Matrix: two-dimensional array

Table with rows and columns: M x N

Three-dimensional array/matrix

a rectangular cuboid of elements: M x N x K.

Multi-dimensional arrays/matrices...

All these arrays are simply *the same* data-type in Matlab, with just *different dimensions*, or size.

MATLAB components: library

>> `myVar(2,3) = 5;` assign values to a variable directly to its place in the array
myVar =

```
0 0 0
0 0 5
```

>> `myVar([1 2],3) = [5 6];`
myVar =

```
0 0 5
0 0 6
```

>> `x(2) = y;` assign a value of one variable to another variable:

`myVec = [1 2 3 4];` Delete an element from an array:

`myVec(2) = [];`

`myVec([2 3]) = [];`

Let's try it out

MatlabBasic.m: Example 5-6

```

% Example 5 - Matrices
myMatrix = [5 3 7 -2; 8 9 1 4; -25 200 34 72]
matrixElementRetrieval = myMatrix(2,3)
% invalidMatrix = [2 4 6; 1 3] % error - why?
arrayRetrieval = myMatrix(1, [2 4])

% Special indexing by 'end' can be used in matrices as well
% seriesMatrixRetrieval = myMatrix([1 : end], [2 : end])

% size/length of matrices
size(myMatrix)
length(myMatrix)

% Replacing rows
myMatrix([1 2], :) = myMatrix([2 1], :)

% Deleting elements
myMatrix(:, 3) = []
size(myMatrix)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Example 6 - Special matrices

myZeros = zeros(2,3)
% myOnes = ones(3,2)
myRandom = rand(2,2) % uniform
distribution
% myNormalRandom = randn(3,3) % normal
distribution

```

MATLAB: text variable

Text variables are comprised of characters and marked with “

```
>>myChar = 'h';
```

```
>>myChar = '5'; (this is not number 5 but the character 5)
```

A text variable can contain more than one **character-string** (an array of characters).

```
>>firstString = 'hello';
```

```
>>secondString = 'world';
```

```
>>longerOne = 'hello world';
```

```
>>longerOne2 = [firstString secondString];
```

(what's wrong with that?)

Let's try it out

MatlabBasic.m: Example 7

```
% Example 7 - Strings

% Create a string
shortString = 'Keep smiling' % note the workspace

% Check the size and length of the string
size(shortString)
length(shortString)

% Concatenation
longString = [shortString 'all day'] % --> Not OK
longString = [shortString ' all day'] % overrides previous
definition of longString

% Check if the strings are the same
isEqual = strcmp(shortString, longString)

% Retrieve only the second part
secondPart = longString(13:end)
% or:
% secondStrIndx = length(shortString)+1 ;
% secondPart = longString(secondStrIndx:end)

% Check if equal
isEqual = strcmp(secondPart, 'all day') % Why are they not equal?
```


Practice 2

In the command window, do the following:

- Create a text variable that contains one word.

- Create another text variable that contains one or more words.

- Concatenate the two strings to create a third variable.

- Display one of the strings in the command window using 'disp' function.

MATLAB: other basics

Functions:

Matlab has a HUGE number of ready-to-use functions/commands.

Examples: length, size, pwd, sum, mean, std, zeros, rand, randn, save, load, and many more...

Files:

Matlab code files have a '.m' extension and include lines of code.

Help:

Matlab has a great help function

```
>> help name_of_function
```

Practice 3

Create a Matlab code file and save it in your current directory.

In this file, do the following:

- Create a 3x4 matrix with values as you like.

- Change the value of the element in the 2nd row and 3rd column.

- Change all the values in the 2nd column at once by assigning a new vector.

- Swap columns 1 and 3.

- Delete the 4th column.

- Use 'size' function to check for the size of the matrix.

Keep practicing ...

```
>>Image=imread('picture.jpg')
```

```
>>imshow(Image)
```

```
>>size(Image)
```

```
756    1134    3
```

Color image:

3D Matrix of RGB planes

```
I=Image;
```

Show RED plane:

```
>>I(:, :, 2:3) = 0;
```

```
>>imshow(I);
```

Show image in black and white:

```
>> bw=rgb2gray(I)
```

```
>> imshow(bw)
```

Try to crop image

