Matlab introduction- supercalculator-essentials

Practicals

- Course certificate will become available in your KU Loket (sign the attendance list)
- https://admin.kuleuven.be/icts/opleidingen/cursusmateriaal
    - Copies of the slides
    - Zip file with sample programs
    - Screencasts
- No food and drinks in the computer room
    - Snack zone available

Overview

Aim

- Help users become adept at using MATLAB as a super calculator.
- Introduce users to the basics of MATLAB

Topics covered

- Introduction to MATLAB
- MATLAB Desktop
- Creating Arrays
- Array Operations
- Logical Operations
- M-file script basics
- Live editor
- IO on a high level
- Create graphics interactively
- Basic Mathematical Manipulations

Introduction

How to use Matlab?

- Interactive mode
  - type commands and use/define variables in command window
- Program
  - Simple scripts
    - M-file (name.m) with list of commands
    - Operate on existing data in work space, or create new data
    - Variables remain in workspace (until cleared)
  - M-file functions
    - M-file as with scripts
    - May return values
    - Easy to call from other functions (make sure file is in MATLAB search path)

Main parts:

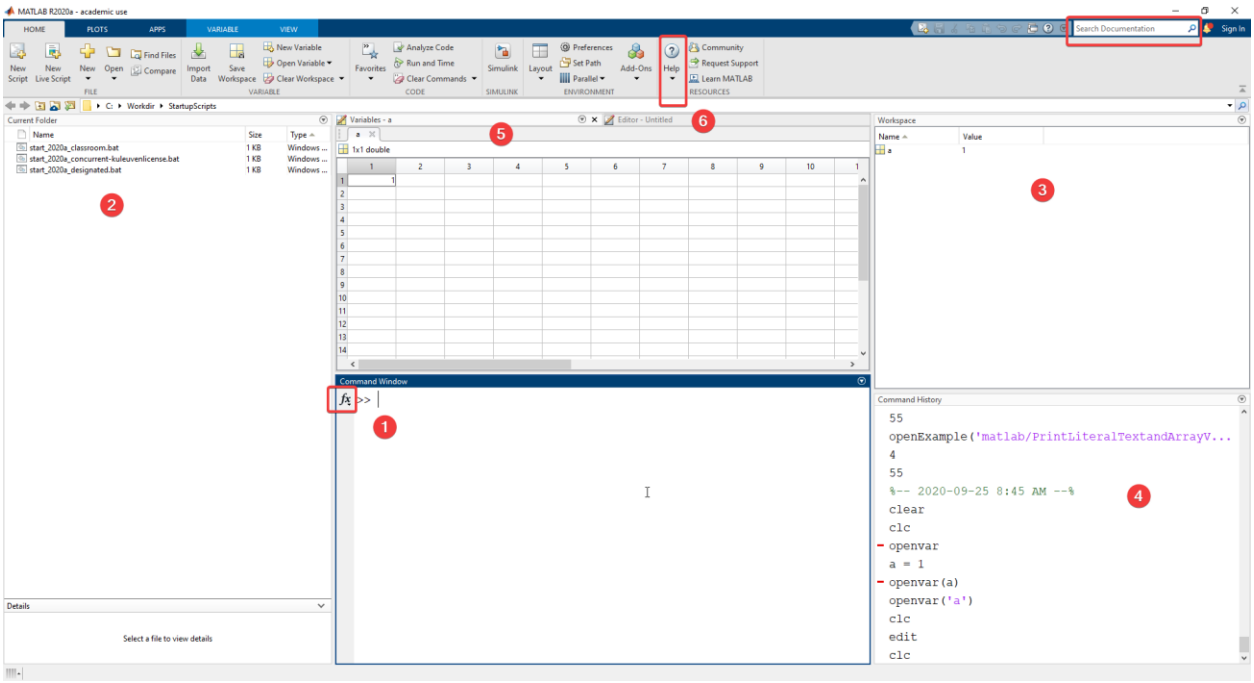
1. Desktop tools and development environment
   a. Mainly graphical user interfaces, editor, debugger, and workspace
2. Mathematical function library
   a. Basic math functions such as sums, cosine, complex numbers
   b. Advanced math functions such as matrix inversion, matrix eigenvalues, differential equations
3. The language
   a. High-level language based on arrays, functions, input/output, and flow statements (for, if, while)
4. Graphics
   a. Data plotting in 2d and 3d,
   b. Image analysis and animation tools
5. External interfaces
   a. Interaction between MATLAB and other programming languages: C, FORTRAN, Java, Python

https://www.mathworks.com/

https://www.tiobe.com/tiobe-index/

https://blogs.mathworks.com/

Matlab Desktop



MATLAB desktop:

- Command Window: enter commands and see output
- Current Folder: directory browser
- Workspace: shows the variables currently in memory
- Command History: list past commands
- Variable Editor: inspect and edit the variables in the workspace
- Editor: edit m files (scripts, functions)

Help in Matlab

- Click on Help-icoon or `F1`
- `help` + *command name*: online help in Command Window
- `doc` + *command name*: online documentation
- `lookfor` + *search string*: search for search string in MATLAB path

Useful commands

- `clc` : clear command window
- `clear` : clear variables and functions from the workspace
- `ctrl + c` : stop the execution of MATLAB
- `whos` : list current variables
- `close` : close the current figure
- `figure` : create a new figure window
- `%` : comment
- `F9`: execute selected command

- ; suppress output to the screen
- `Alt`: display hotkeys
- `Tab-key`: auto completion

Predefined variables

- `ans` : The default variable name when no variable has been specified.
- `pi` : π.
- `eps` : the smallest positive real number on the computer such that 1 + eps ~= 1.
- `Inf` : ∞ (as in 1/0).
- `NaN` :Not-a-Number (as in 0/0).
- `i, j` : √(−1)

Variables / Assignment

Variable names consist of a letter, followed by any number of letters, digits, or underscores.

- MATLAB uses only the first 63 characters of a variable name.
- MATLAB is case sensitive

```
A = 8
```

```
a = 8
```

Array creation

Arrays

- Creating **[***how to build an array in a fast way***]**
  - manually entering the elements
    - `v = [1, 2, 3]    (row vector)`
    - `v = [1; 2; 3]    (column vector)`
    - `A = [1, 2; …     (… continuation )`
      `2, 3]`
  - Sequence generation **:**
    - `v = [1:10]`
  - `linspace, logspace`
    - `v = linspace(0,1,11)`
  - special functions, special matrices
    - `size`: determine the size
    - always keep track of the dimensions of your arrays
  - concatenate
    - `C = [A; B]  or C = [A B]`

- Indexing **(***how to select elements***)**
  - `v(3)`
  - `v(1:3), v(1:2:end), v(end)`
  - `A(1,1), A(n,m), A(end, end)`
  - `A(n,:)` select the nth row
  - `A(:,m)` select the mth column
  - `A(1:end)` convert the array into a vector

Aray operations

	function	
`A+B`	`plus(A,B)`	Binary addition
`A-B`	`minus(A,B)`	Binary subtraction
`A*B`	`mtimes(A,B)`	Matrix multiplication
`A.*B`	`times(A,B)`	Arraywise multiplication
`A/B`	`mrdivide(A,B)`	Matrix right division *Divide by* post-multiplication by the inverse of a matrix $A*B^{-1}$
`A./B`	`rdivide(A,B)`	Arraywise right division
`A\B`	`mldivide(A,B)`	Matrix left division *Divide into* The solution to Ax = b for A ∈ C m×n : when m = n and A is nonsingular this is the solution Gaussian elimination; when m > n this is the least-squares approximation of the overdetermined system; when m < n this is a solution of the underdetermined system pre-multiplication by the inverse of a matrix $A^{-1}*B$
`A.\B`	`ldivide(A,B)`	Arraywise left division
`A^B`	`mpower(A,B)`	Matrix power
`A.^B`	`power(A,B)`	Arraywise power
`A'`	`ctranspose(A)`	Complex transpose
`A.'`	`transpose(A)`	Matrix transpose

Most functions operate on columns by default.

Relational operators and logical operators

Relational Operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Logical Operators

```
A = [0 1 1 0 1];
B = [1 1 0 0 1];
```

Operator	Description	Example
&	Returns 1 for every element location that is true (nonzero) in both arrays, and 0 for all other elements.	A & B = 01001
|	Returns 1 for every element location that is true (nonzero) in either one or the other, or both, arrays and 0 for all other elements.	A | B = 11101
~	Complements each element of input array, A.	~A = 10010

Tip: use parentheses

Function	Description
`any(x)`	True if any element in vector x is nonzero. True for each column in a matrix x that has nonzero elements.
`all(x)`	True if all elements in a vector x are nonzero. True for each column in a matrix that has all nonzero elements.

A lot of is-functions

Tip:

- use the autocomplete feature
- use the function browser 

function	
`isfinite(x)`	True where elements of x are finite
`isinf(x)`	True where elements of x are infinite
`islogical(x)`	True if x is a logical array
`isnumeric(x)`	True if x is a numeric array

Logical indexing

1. Use conditional operators to create a logical array of the same size as the original.
2. Use logical array to pick out indices that satisfy conditions.

Create plots interactively

Overview of different plot types + example code: https://nl.mathworks.com/products/matlab/plot-gallery.html

Chart chooser: https://blogs.mathworks.com/videos/2009/01/16/flow-chart-shows-which-visualization-to-use/

Steps:

Prepare your data
Call elementary plotting function
Select line and marker characteristics
Set axis limits, tick marks, and grid lines marks, and grid lines
Annotate the graph with axis labels, legend, text and title

Brackets, Parentheses, and Curly Braces

Copied from http://nens230.stanford.edu/braces.html

Matlab syntax uses a combination of brackes [], parentheses (), and curly braces {}. It can often be confusing when it is appropriate to be using which symbols.

Parentheses ( )

Parentheses are used for:

Indexing into an array	`x(1:3)`
Defining order of operations	`(3+4)^2`
Function inputs	`mean(x)`

Brackets [ ]

Brackets are used to:

Create an array or matrix	`x = [1 2; 3 4]`
Delete (excise) elements	`x(x < 0) = []`
Group function outputs	`[value index] = max(x)`

Curly Braces { }

Curly braces are used to:

Create a cell array	`bases = {'A', 'G', 'T', 'C'}`
Get content from a cell array	`guanine = labels{2}`