FF505

Computational Science

MATLAB Section - Introduction 1

Matrix Algebra

Marco Chiarandini (marco@imada.sdu.dk)

Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark

Outline

Outline

MATLAB Desktop

- Command window
- Workspace
- Command history
- Current folder browser
- Variable editor
- MATLAB program editor
- Help
- Desktop menu
- Docking/Undocking, maximize by double click
- Current folder
- Search path (`Set Path`)
- Documentation: Press ? → MATLAB → Getting Started

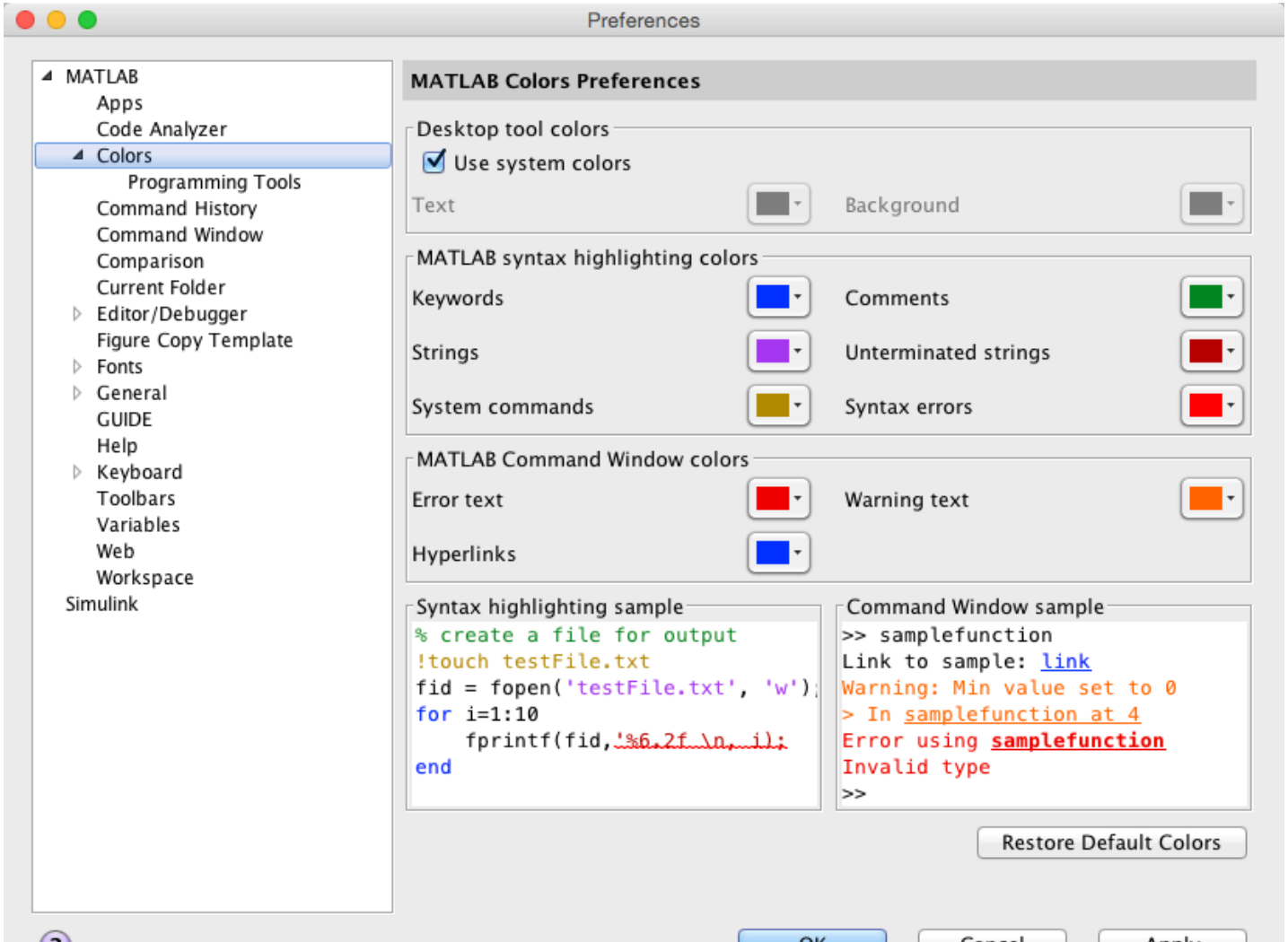
Command line programming

```
%%% elementary operations
5+6
3-2
5*8
1/2
2^6
1 == 2 % false
1 ~= 2 % true. note, not "!="
1 && 0
1 || 0
xor(1,0)
```

Customization

`MATLAB -> preferences`

Allows you personalize your MATLAB experience



Outline

Variables

Different meaning than in mathematics, ie, not unknowns in a formula but:

- a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.
- separation of name and content
- compilers have to replace variables' symbolic names with the actual locations of the data.
- a variable's name, type, and location often remain fixed, while the data stored in the location may be changed during program execution.
- in mathematics typically identified by single letters, in computer programming long, explanatory names are preferred.
- The scope of a variable is the portion of the program code for which the variable's name has meaning and for which the variable is said to be "visible". global vs local variables.
- A variable whose scope begins before its extent does is said to be uninitialized and often has an undefined, arbitrary value!

Variable Assignment

The = sign in MATLAB represents the assignment or replacement operator. It has a different meaning than in mathematics.

Compare:

`x = x + 3` In math it implies 0=2, which is an invalid statement
In MATLAB it adds 2 to the current value of the variable

```
%% variable assignment
a = 3; % semicolon suppresses output
b = 'hi';
c = 3>=1;

% Displaying them:
a = pi
disp(sprintf('2 decimals: %0.2f', a))
disp(sprintf('6 decimals: %0.6f', a))
format long % 16 decimal digits
a
format short % 4 decimal digits +
     scientific notation
a
```

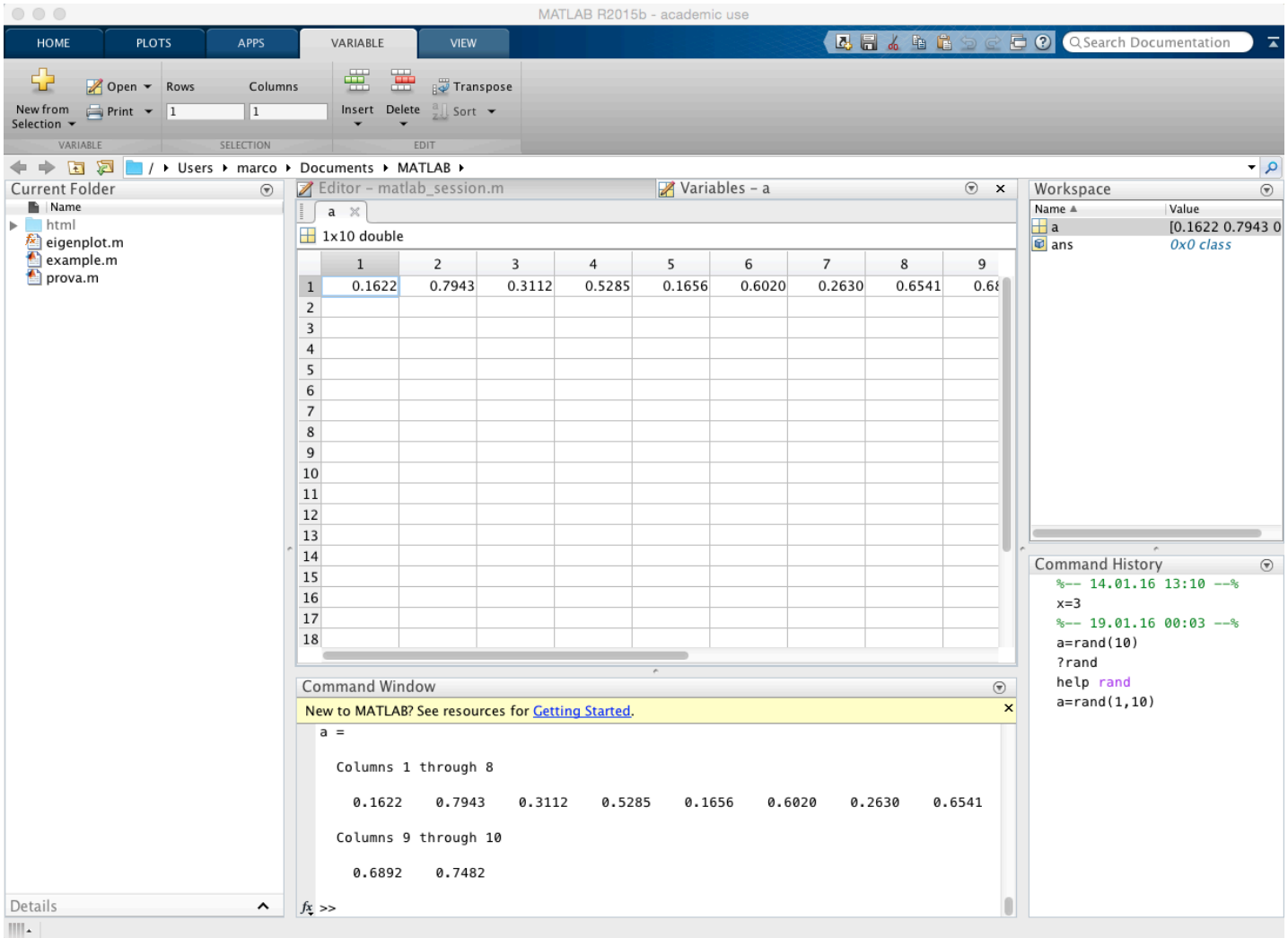
```
x + 2 = 20 % wrong statement
x = 5 + y % wrong if y unassigned
```

Variables are visible in the workspace

Names:

- [a-z] [A-Z] [0-9] _
- case sensitive
- max 63 chars

Variable Editor



Outline

Managing the Work Session

```
who % lists variables currently in memory
whos % lists current variables and sizes
clear v % clear w/ no argt clears all
edit filename % edit a script file
clc % clears theCommand window
... % ellipsis; continues a line
help rand % returns help of a function
quit % stops MATLAB
```

Predefined variables

```
pi
Inf % 5/0
NaN % 0/0
eps % accuracy of computations
i,j % immaginary unit i=j=sqrt(-1)
3+8i % a complex number (no *)
Complex(1,-2)
```

Working with Files

MATLAB handles three types of files:

- M-files `.m`: Function files, program files and scripts
- MAT-files `.mat`: binary files with name and values of variables
- data file `.dat`: ASCII files

```
%% loading data
load q1y.dat
load q1x.dat
save hello v; % save variable v into file
     hello.mat
save hello.txt v -ascii; % save as ascii
% fopen, fprintf, fscanf also work
% ls %% cd, pwd & other unix commands
      work in matlab;
% to access shell, preface with "!"
```

Files are stored in the current directory and searched search path

Directories and paths

If we type `problem1`

1. seeks if it is a variable and displays its value
2. checks if it is one of its own programs or functions and executes it
3. looks in the current directory for file `program1.m` and executes the file
4. looks in the search path for file `program1.m` and executes it

```
addpath dirname % adds the directory dirname to the search path
cd dirname % changes the current directory to dirname
dir % lists all files in the current directory
dir dirname % lists all files in dirname
path % displays the MATLAB search path
pathtool % starts the Set Path tool
pwd % displays the current directory
rmpath dirname % removes the directory dirname from the search path
what % lists MATLAB specific files in the current directory
what dirname % lists MATLAB specific files in dirname
which item % displays the path name of item
```

Getting Help

- `help funcname`: Displays in the Command window a description of the specified function `funcname`.
- `lookfor topic`: Looks for the string topic in the first comment line (the H1 line) of the HELP text of all M-files found on MATLABPATH (including private directories), and displays the H1 line for all files in which a match occurs.
  Try: `lookfor imaginary`
- `doc funcname`: Opens the Help Browser to the reference page for the specified function `funcname`, providing a description, additional remarks, and examples.
- When typing a function a hint (tooltip) shows the list of arguments.

Outline

Scripts, M-files
Overview

Scripts are

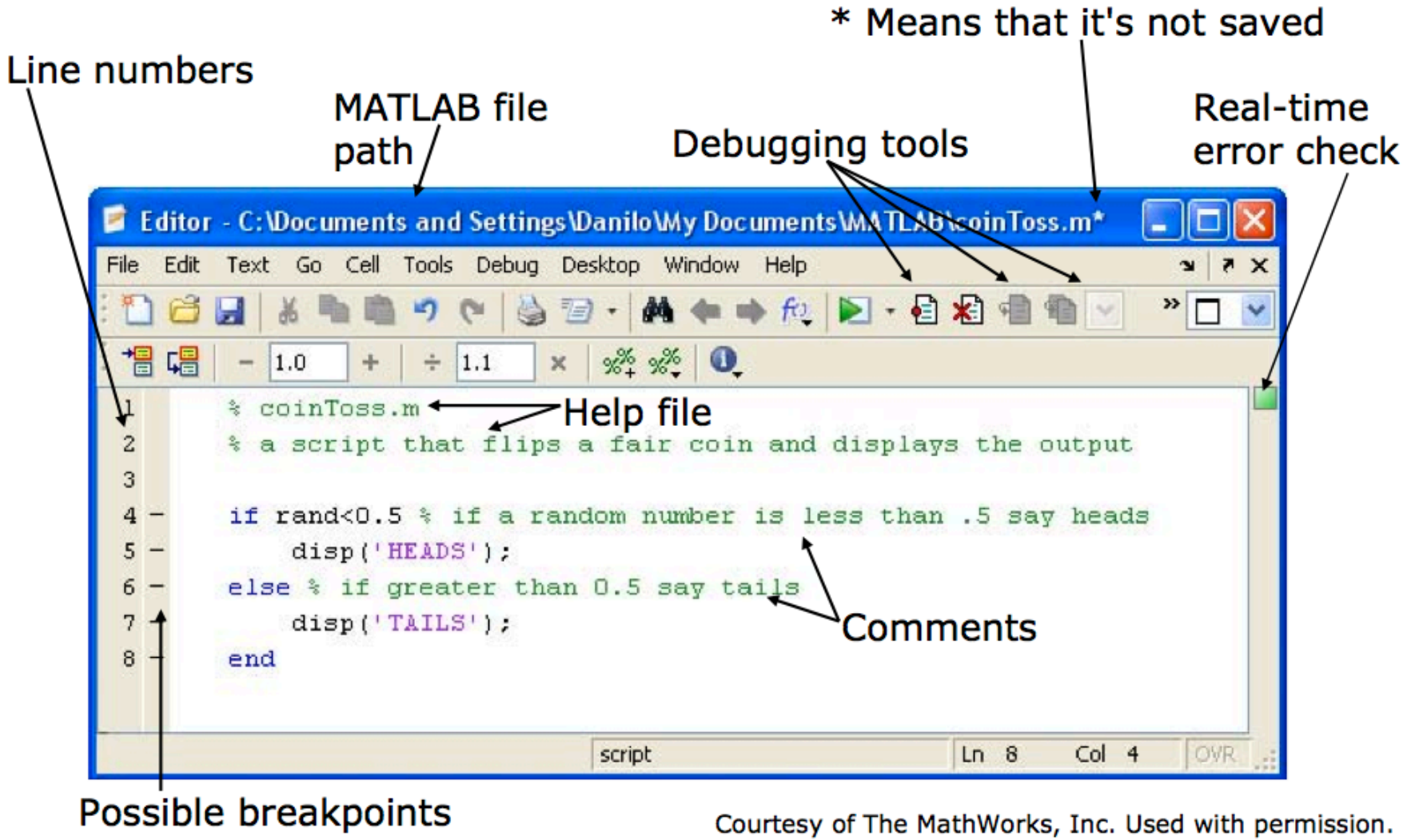
- collection of commands executed in sequence
- written in the MATLAB editor
- saved as MATLAB files (.m extension)

To create a MATLAB file from command-line

```
edit helloWorld.m
```

or from Menu on the top

Script: the Editor



Courtesy of The MathWorks, Inc. Used with permission.

Exercise: Scripts

- Make an initial script `ost` and save it.
- When run, the script should display the following text:

```
This is my first script! Yuhuu!
```

Hint: use `disp` to display strings. Strings are written between single quotes, like `'This is a string'`

Outline

1-D Arrays

Vectors: To create a row vector, separate the elements by commas. Use square brackets. For example,

```
>> p = [3,7,9]
p =
   3 7 9
```

You can create a column vector by using the transpose notation (').

```
>> p = [3,7,9]'
p =
   3
   7
   9
```

You can also create a column vector by separating the elements by semicolons. For example,

```
>> g = [3;7;9]
g =
   3
   7
   9
```

Appending vectors:

```
r = [2,4,20];
w = [9,-6,3];
u = [r,w]
u =
    2 4 20 9 -6 3
```

```
r = [2,4,20];
w = [9,-6,3];
u = [r;w]
u =
    2 4 20
    9 -6 3
```

2-D Arrays

Matrices: spaces or commas separate elements in different columns, whereas semicolons separate elements in different rows.

```
>> A = [2,4,10;16,3,7]
A =
     2 4 10
    16 3 7

>>c = [a b]
c =
   1 3 5 7 9 11

>>D = [a ; b]
D =
   1 3 5
   7 9 11
```

Arrays

Arrays are the basic data structures of MATLAB (weakly typed language - no need to declare the type)
Types of arrays:
numeric • character • logical • cell • structure • function handle

```
%% vectors and matrices
A = [1 2; 3 4; 5 6]

v = [1 2 3]
v = [1; 2; 3]
v = [1:0.1:2] % from 1 to 2, with stepsize of 0.1. Useful for plot axes
v = 1:6 % from 1 to 6, assumes stepsize of 1

C = 2*ones(2,3) % same as C = [2 2 2; 2 2 2]
w = ones(1,3) % 1x3 vector of ones
w = zeros(1,3)
w = rand(1,3) % drawn from a uniform distribution
w = randn(1,3) % drawn from a normal distribution (mean=0, var=1)
w = -6 + sqrt(10)*(randn(1,10000)) % (mean = 1, var = 2)
hist(w) % histogram
e = []; % empty vector
I = eye(4) % 4x4 identity matrix
A = linspace(5,8,31) % equivalent to 5:0.1:8
```

Indexing

```
%% indexing
A(3,2) % indexing is (row,col)
A(2,:) % get the 2nd row. %% ":" means every elt along that dimension
A(:,2) % get the 2nd col
A(1,end) % 1st row, last elt. Indexing starts from 1.
A(end,:) % last row

A([1 3],:) = [] % deletes 1st and 3rd rows
A(:,2) = [10 11 12]' % change second column
A = [A, [100; 101; 102]]; % append column vec
% A = [ones(size(A,1),1), A]; % e.g bias term in linear regression
A(:) % Select all elements as a column vector.
```

```
%% dimensions
sz = size(A)
size(A,1) % number of rows
size(A,2) % number of cols
length(v) % size of longest dimension
```

Outline

Plots

```
%% plotting
t = [0:0.01:0.98];
y1 = sin(2*pi*4*t);
plot(t,y1);
y2 = cos(2*pi*4*t);
hold on; % "hold off" to turn off
plot(t,y2,'r--');
xlabel('time');
ylabel('value');
legend('sin','cos');
title('my plot');
close; % or, "close all" to close all figs
```

```
figure(2), clf; % can specify the figure number
subplot(1,2,1); % Divide plot into 1x2 grid, access 1st element
plot(t,y1);
subplot(1,2,2); % Divide plot into 1x2 grid, access 2nd element
plot(t,y2);
axis([0.5 1 -1 1]); % change axis scale
```

```
help graph2D
```

Rapid Code Iteration

- Rapid code iterations using cells in the editor
- cells are small sections of code performing specific tasks
- they are separated by double %
- they can be executed independently, eg, CTRL+Enter and their parameters adjusted
- navigate by CTRL+SHIFT+Enter or by jumping
- publish in HTML or PDF or Latex (menu publish on the top).

Summary

1. Getting Started
2. Variables
3. Work session
4. Scripts
5. Arrays
6. Plotting